Continual Robot Learning Using Self-Supervised Task Inference

Muhammad Burhan Hafez and Stefan Wermter, *Member, IEEE*

***Abstract*—Endowing robots with the human ability to learn a growing set of skills over the course of a lifetime as opposed to mastering single tasks is an open problem in robot learning. While multitask learning approaches have been proposed to address this problem, they pay little attention to task inference. In order to continually learn new tasks, the robot first needs to infer the task at hand without requiring predefined task representations. In this article, we propose a self-supervised task inference approach. Our approach learns action and intention embeddings from self-organization of the observed movement and effect parts of unlabeled demonstrations and a higher level behavior embedding from self-organization of the joint action–intention embeddings. We construct a behavior-matching self-supervised learning objective to train a novel task inference network (TINet) to map an unlabeled demonstration to its nearest behavior embedding, which we use as the task representation. A multitask policy is built on top of the TINet and trained with reinforcement learning to optimize performance over tasks. We evaluate our approach in the fixed-set and continual multitask learning settings with a humanoid robot and compare it to different multitask learning baselines. The results show that our approach outperforms the other baselines, with the difference being more pronounced in the challenging continual learning setting, and can infer tasks from incomplete demonstrations. Our approach is also shown to generalize to unseen tasks based on a single demonstration in one-shot task generalization experiments.**

***Index Terms*—Continual multitask learning, robot control, self-supervised learning, task inference.**

I. Introduction

TEACHING robots to perform tasks with minimal knowledge about the environment and without manually programming the desired behavior has always been the driving motivation for the research on robot learning. The field has witnessed remarkable progress over the past decade on a variety of difficult tasks, including manipulation [1], [2], locomotion [3], and navigation [4]. However, the learning is more oriented toward solving single tasks. To enable multitask learning, approaches based on knowledge distillation [5], [6], [7], metalearning [8], [9], and language conditioning [10], [11] have been proposed.

Manuscript received 22 December 2022; revised 13 May 2023 and 24 July 2023; accepted 9 September 2023. Date of publication 14 September 2023; date of current version 11 June 2024. This work was supported by the German Research Foundation DFG under Project CML (TRR 169). *(Corresponding author: Muhammad Burhan Hafez.)*

The authors are with the Knowledge Technology Group, Department of Informatics, University of Hamburg, 22527 Hamburg, Germany (e-mail: burhan.hafez@uni-hamburg.de; stefan.wermter@uni-hamburg.de).

Digital Object Identifier 10.1109/TCDS.2023.3315513

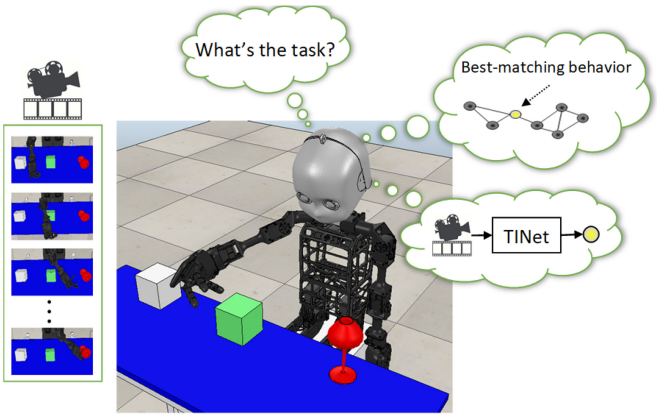

Fig. 1. Given an unlabeled demonstration (left), the robot learns to infer the task at hand (e.g., *Grasp the red glass*) by finding the best-matching behavior in the growing, self-organizing network of behaviors and training the proposed TINet to map the demonstration to its best-matching behavior, which we use as the task representation.

While these approaches are becoming widely adopted to overcome the limitation of task-specific learning, they have two notable deficiencies. First, they are not compatible with continual learning as they assume a fixed task distribution and treat newly introduced tasks as independent learning problems. Specifically, when a policy to perform a given set of sensorimotor tasks is learned, a complete retraining is often required if a new task is introduced. This is in sharp contrast to the human ability to learn a growing repertoire of skills over the course of a lifetime. Second, they lack an efficient task inference mechanism. This issue is either ignored by using predefined task labels or fixed task representations (e.g., pretrained natural language embeddings) as input or poorly addressed by requiring extensive exploration to gather experience data sufficient to infer a posterior over a latent task variable [12], [13]. Humans, on the other hand, only need to observe a demonstration of the desired behavior to successfully infer the task at hand due to their ability to understand and imitate the goal of the observed behavior, not the precise actions [14], [15].

Existing multitask learning approaches that incorporate expert demonstrations can be categorized, according to the way the demonstrations are used, into three distinct groups: 1) conditioning the policy on a demonstration embedding [16], [17]; 2) training the policy to match demonstration actions with behavior cloning [18], [19]; and 3) generating a reward based on how close the observed image is to the corresponding one from visual demonstrations [20], [21]. A

common assumption in these groups of approaches is that the demonstrations are complete. This is a strong assumption in practice for several reasons, such as the misalignment between the initial state of the demonstration and that of the robot's environment, sensory noise, self- and object-occlusion, and motion blur. Consider the following example as motivation: if a demonstration of some desired visual manipulation task has missing or corrupted parts (e.g., irretrievable image observations) for any of the above-mentioned reasons, then the control policy can only learn to match the observed actions of the intact part of the demonstration without the ability to recover the remaining actions. In the case of using this demonstration to infer or identify the desired task before solving it by conditioning the policy on the demonstration, the policy will be unable to infer the correct task, and much worse, using this as a training example will impair the learned policy. A multitask learning approach that enables task inference from incomplete demonstrations is thus needed to relax this assumption. Moreover, such an approach would be consistent with a large body of behavioral and neurocognitive evidence indicating that human children can imitate incomplete task demonstrations [22], [23], [24], [25], [26], [27]. Besides assuming complete demonstrations, the above groups treat each demonstration as one single unit of information, overlooking the fact that a demonstration is a combination of action, which is the observed movement, and intention, which is the observed effect. In contrast, learning movement-effect associations by observation has been found to play an essential role in the developmental changes in human goal-directed imitation, including the ability to imitate behaviors from incomplete demonstrations [25], [28], [29].

In this article, we propose a multitask robot learning approach that alleviates the two deficiencies identified earlier—namely, the incompatibility of the existing approaches with continual learning of sensorimotor tasks and the lack of efficient task inference mechanisms. Our approach learns, in an unsupervised manner, a behavior embedding space from unlabeled task demonstrations. We construct a behavior-matching self-supervised learning objective for training a novel task inference network (TINet) to map a given demonstration to its nearest behavior embedding, which we use as the task representation (see Fig. 1). A multitask policy is built on top of the TINet and trained with reinforcement learning (RL) to optimize performance over tasks.

In a previous work [16], incremental self-organization of visual demonstrations of behaviors was proposed to build a behavior embedding space for efficient task inference in continual robot learning. However, a single self-organizing network was used to learn to map an unlabeled demonstration to a behavior embedding. This means that the network will map an incomplete demonstration to a behavior embedding different from the one that best matches the complete demonstration. Furthermore, a node representing this undesired behavior will be added to the network if the node insertion criterion is met, which impairs the behavior embedding space. In contrast, using two networks separately learning actions and intentions and another learning action–intention associations facilitates finding a correct behavior because from the intention embedding of an incomplete demonstration it will be possible to retrieve a behavior that has the same or similar intention as the one that best matches the complete demonstration. We improve on [16] by treating the visual demonstration as a combination of an observed movement and an observed effect and learning two separate embeddings for each of these two components of a demonstration, which we call action and intention embeddings, respectively. The behavior embedding space is learned by incrementally self-organizing the combined action and intention embeddings. Unlike [16], our approach can perform task inference from incomplete demonstrations. This is achieved by randomly sampling a subtrajectory from the demonstrated trajectory and training the proposed TINet to map both trajectories to the behavior embedding that best matches the demonstrated trajectory with a behavior-matching self-supervised learning objective. Furthermore, the whole learning architecture, including the policy network and the TINet, is trained end-to-end which allows the task representations to capture the structure of the task at hand.

The primary contributions of our work are summarized as follows.

1) We develop a hierarchical architecture to learn unsupervised embeddings of actions, intentions, and the resulting action–intention associations from unlabeled demonstration data.
2) A behavior-matching self-supervised learning objective is proposed to train a TINet to map an input demonstration to the best matching behavior in the unsupervisedly learned behavior embedding space.
3) We introduce an end-to-end continual robot learning approach that learns novel tasks over time and can infer tasks from incomplete visual demonstrations.
4) We evaluate our approach in multitask learning experiments under the continual learning setting with a humanoid robot and compare it to different multitask learning baselines.

II. Related Work

A. Task-Agnostic Models and Skills

It has been shown that a world model learned by unsupervised exploration can be used to efficiently solve multiple tasks [30]. First, a task-agnostic exploration policy is trained to collect experience that improves the world model by maximizing expected novelty of future states. A task policy is then trained by imagination inside the world model. The method is found to achieve better zero-shot task performance than other unsupervised methods on continuous control tasks and few-shot performance comparable to a supervised oracle that receives task rewards during exploration. However, fast adaptation to a downstream task requires the world model to be trained in parts of the environment relevant to the task, which cannot be guaranteed with the proposed exploration method. Similar to [30], Sharma et al. [31] trained an RL agent without reward supervision and use it to solve downstream tasks. However, the training objective is not to learn a world model, but rather to learn a set of reusable skills that can be composed

when solving a given task. These skills are learned to be diverse by iteratively sampling a skill from a skill prior and encouraging a skill-conditioned policy to produce transitions that are predictable, given the sampled skill, and different from those produced by the policy conditioned on a different skill. At test time, model-predictive control is used to find an optimal sequence of learned skills for solving a target task without any learning on the task. One issue with the proposed method is that the learned skill-conditioned transition model is queried at test time on states generated by the model itself at previous timesteps, which can be different from the state distribution it was trained on.

Another approach extracts reusable skills from an experience data set collected across different tasks and recombines them to efficiently solve a downstream task [32]. A variational encoder computes a skill embedding for each trajectory sampled from the data set and a low-level policy is trained with behavior cloning to decode the embedding into its corresponding actions. State-conditioned skill prior and posterior are trained to match the pretrained skill encoder on behaviors from the experience data set and task demonstrations, respectively. To solve a downstream task, a high-level policy over skill embeddings is trained with RL using an objective that constrains the policy to be close to the skill posterior if the environment state comes from the demonstration data, or to the skill prior otherwise. The approach is shown to outperform prior works that use either demonstrations or task-agnostic experience. However, it makes a strong assumption that the experience data set contains meaningful, short-horizon behaviors and requires training a separate high-level policy from scratch for every new downstream task.

Our approach shares a common objective with this group of approaches, which is to enable fast adaptation to downstream tasks. However, it stands out by not relying on a world model or an experience data set.

B. Learning Task-Conditioned Policies

Lynch et al. [19] proposed a method for learning a continuum of robotic tasks from unlabeled play data. The method learns a latent plan distribution space from play sequences by optimizing for reconstruction of play actions while maximizing the similarity between the latent plan distribution of each sequence and that of its combined initial and final states. At test time, a latent plan is sampled given the current and goal states. It is then fed with the two states to a stochastic policy trained to reconstruct the actions of a play sequence from its corresponding latent plan, initial, and final states. While not requiring expensive expert demonstrations, the proposed method trains the policy on trajectories of play data collected by curious exploration that aims to sufficiently cover the state–action space without regard to the quality of the generated behavior. This leads to a poor policy when the training trajectories are far from the optimal behavior.

To enable zero-shot generalization to novel tasks, Jang et al. [17] proposed to condition the policy on information that describes the task, such as language instruction or video demonstration. A task embedding is computed from this information and passed into the policy which is supervised with behavior cloning to match the actions in the task demonstrations. The video encoder is constrained to produce an output that is close to the pretrained embedding of the corresponding language description to align the videos more semantically. While the trained policy is found to generalize to unseen tasks, the performance of the video-conditioned policy is lower than the language-conditioned one. The method also requires a predefined task data set on which the policy is trained at once, and hence the method is not applicable when tasks are presented over time.

Sodhani et al. [33] found that learning context-based composable representations is an efficient method for sharing information across tasks in multitask RL. In their approach, a natural language task description acts as the context and a mixture of encoders is used to give multiple representations to an input observation. The context determines how to compose the representations by computing soft-attention weights over representations. The weighted sum of representations is concatenated with the context vector and fed to the policy network. Despite improving knowledge transfer, the approach strongly relies on the language description's semantics to extract task-relevant object and skill representations and infer similarity between tasks. It is therefore incompatible with other forms of task description, including visual demonstration.

While our approach, like this group of approaches, conditions the policy on a task description, it is not limited by the quality of task demonstrations or language semantics to extract task-relevant information.

C. Cross-Task Adaptive Regularization

To accelerate learning new tasks while preserving performance on previously learned tasks, Schwarz et al. [34] proposed to use two neural networks: an active column and a knowledge base. The former is used to learn a new task and is layerwise connected with the latter to utilize past information. After a task is learned, the active column is distilled into the knowledge base whose parameters are regularized to be close to those adapted to older tasks. The approach has two limitations when used to train a multitask policy. First, due to the regularization constraint, the knowledge base will learn a policy that tries to achieve average performance on all encountered tasks instead of optimal performance on each individual task. Second, the approach does not address task inference and assumes that changes in task distribution are known to the learner. Hessel et al. [35] suggested that to improve the performance of multitask RL, all tasks should influence the learning updates similarly regardless of the density and scale of their rewards. They propose an actor–critic method that learns multiple tasks in parallel by assigning different environments to different actors. Iteratively, the experience collected by all actors is used to update both a value network with multiple outputs, one for each task, and a task-agnostic policy network used by the actors. The targets in the updates are adaptively normalized by tracking the statistics of the return in each task, causing tasks with different return scales to have

similar impact on the learning. A limitation to the proposed method is that the output layer of the value network depends on a predefined number of tasks, rendering it infeasible for learning new tasks over time. Furthermore, parallel training is resource-intensive and impractical for real robots.

Similar to [34] and [35], our approach trains a multitask policy with RL, but does not require prior knowledge of task distribution or value network reconfiguration when new tasks are added.

D. Leveraging Task Relationships

While most approaches to multitask learning give little consideration to task relationships beyond learning generalizable task features from task examples, a few have shown that exploiting the relationships between tasks leads to fast adaptation to new tasks [16], [21], [36], [37]. These approaches mainly differ in how task relationships are learned. For example, Oh et al. [36] added an analogy-making objective to encourage the task representation to capture task similarities when optimizing performance over tasks. Kalashnikov et al. [21] proposed a distributed multitask RL method in which off-policy data is collected by multiple robots and shared between similar tasks to improve the efficiency of learning each task. This training data is rebalanced between tasks in every training batch when updating the multitask policy. Besides requiring a predefined discrete set of tasks, that does not allow for continual multitask learning, the method also requires to manually decide which tasks are semantically similar in order to share data between them. Another approach is to train a metamapping function that transforms a learned task representation into another one using a training data set of task representation pairs, where all paired tasks are systematically related [37]. This direct exploitation of systematic relationships has shown better adaptation performance than the indirect way of generalizing through language alone. Instead of relying on prior knowledge in terms of pairs of systematically related tasks [21], [37] or predefined task analogies [36], a more recent work [16] learns task relationships unsupervised by continually self-organizing visual demonstrations of tasks so that behaviorally similar tasks are located close to each other. However, the proposed method makes a strong assumption that task demonstrations are perfect and complete, which is restrictive and not often realistic in practice.

The approach described in this article exploits task relations and learns them in an unsupervised manner from unlabeled task demonstrations, similar to [16]. The difference is that our approach does not assume completeness of the demonstrations, which makes it more robust and applicable in real-world scenarios where complete demonstrations may not be available.

III. Technical Approach

In this section, we present our self-supervised task inference approach for continual multitask robot learning. We start by describing how action and intention embeddings are learned in an unsupervised manner from unlabeled task demonstrations and used to learn behavior embeddings. Then, the TINet is introduced, which is trained with the proposed behavior-matching self-supervised learning objective. Finally, we show how a multitask policy can be trained end-to-end with RL on top of TINet to optimize performance over tasks.

The aim is to train a multitask policy with RL that can recognize the desired task from an incomplete demonstration and successfully execute the task. At the start of every learning episode, a complete demonstration in the form of a trajectory of n images is randomly sampled and encoded into a vector h_n, as shown in Fig. 2. Similarly, the sequence of the first $n-1$ images is encoded into a vector h_{n-1} and the last image is encoded into a vector x_{n-1}. The vectors h_{n-1} and x_{n-1} are used as input to the growing self-organizing networks Action Net and Intention Net, respectively, and the action and intention embeddings g^{act} and g^{int} that best match h_{n-1} and x_{n-1} are identified before the two networks are updated (Section III-A). The combined action–intention embedding in turn is used as input to the growing self-organizing network Behavior Net, where the behavior embedding g^b that best matches the input is identified and the network is updated. The encoded demonstration h_n is fed to the TINet that outputs the task representation z_i. The feature vector of the current environment state s together with z_i are fed to the multitask policy which outputs the action to take. The TINet is trained with contrastive learning to output the same task representation for the input demonstration (complete demonstration) and for a randomly chosen part of the input demonstration (incomplete demonstration) (Section III-B). It is also jointly trained to minimize the distance between its output and the behavior embedding g^b.

A. Hierarchical Self-Organization of Behaviors

Demonstration Encoding: In our approach, a task demonstration is defined as a trajectory of image observations showing the robot performing a particular behavior to complete the desired task. Any behavior can typically be described by different demonstrations, each being a different trajectory that shows a successful completion of the same task. The image observations in a trajectory are encoded by a convolutional neural network (CNN), and the sequence of the CNN encodings is processed by a recurrent neural network based on the long short-term memory (LSTM) architecture [38] to capture the contextual information in the demonstration. We use the hidden state h_{n-1} of the LSTM after the last CNN encoding $x_{n-1} = f_x(T_{n-1})$ has been read as the latent representation of the entire demonstration, where n is the length of the demonstration. Together, the observation-encoding CNN and context-encoding LSTM define a demonstration encoder f_d that takes in a trajectory of observations $T_{0:n-1}$ and outputs a latent representation of the demonstration. Fig. 2(a) illustrates the demonstration encoding process. Section III-C describes how the demonstration encoder is trained. Details on the design choices of the CNN and LSTM networks are given in Section IV.

Action and Intention Embeddings: Each task demonstration is a combination of action and intention, which are the observed movement and effect, respectively. We explicitly

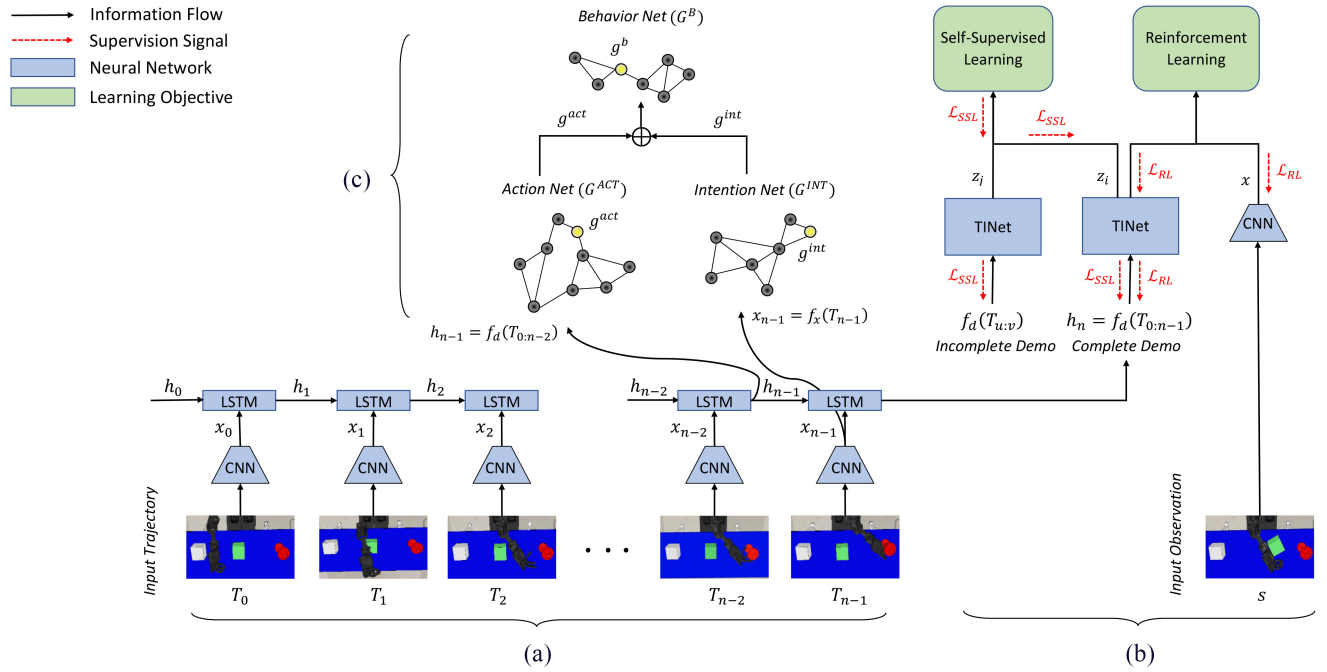

Fig. 2. Overview of our proposed task inference architecture for continual multitask learning. (a) Each input demonstration is encoded with a demonstration encoder f_d by passing image observations in the demonstrated trajectory $T_{0:n-1}$ to a CNN encoder f_x and processing the sequence of CNN-encoded image features $x_{0:n-1}$ with an LSTM. The hidden state h_n after the last feature vector x_{n-1} has been read is used as a latent representation of the demonstration. (b) TINet is trained with a behavior-matching self-supervised learning objective to map a complete $f_d(T_{0:n-1})$ and an incomplete $f_d(T_{u:v|0\leq u<v\leq n-1})$ version of an input demonstration to a behavior embedding g^b in the self-organizing Behavior Net G^B that best matches the input demonstration. The task representation z_i produced by the TINet is used together with the current observation's feature vector $x = f_x(s)$ as input to a multitask policy trained with RL to optimize performance over tasks. (c) Action and intention embeddings are learned in an unsupervised manner by incrementally self-organizing the movement $f_d(T_{0:n-2})$ and effect $f_x(T_{n-1})$ components of input demonstrations using the growing Action Net G^{ACT} and Intention Net G^{INT}, respectively. Given an input demonstration, the action g^{act} and intention g^{int} embeddings that best match the demonstrated movement and effect are combined. The behavior embeddings are in turn learned by self-organizing the combined action and intention embeddings with the Behavior Net G^B.

leverage this fact and learn two mappings: the first maps from an input space of visually described movements to an embedding space where similar movements are located together; the second maps from an input space of visually described effects to an embedding space where similar effects are located together. These embeddings are called action and intention embeddings, respectively. In our approach, both mappings are learned in an unsupervised manner by incrementally self-organizing the respective input space with a growing self-organizing network. Particularly, we use the grow when required (GWR) network [39], which grows when it does not have a close enough match to an input stimulus as opposed to adding nodes at predefined intervals, a criterion often used in other growing networks. This allows adding novel actions and intentions to the respective network once discovered. We refer to the GWR networks used to learn the action and intention embeddings by G^{ACT} and G^{INT}, respectively. For each input demonstration, we pass the first $n-1$ observations to the demonstration encoder f_d whose output $f_d(T_{0:n-2})$ is used as input to G^{ACT} and use the CNN-encoded feature vector x_{n-1} of the last observation as input to G^{INT} [see Fig. 2(c)].

The GWR network is defined by a set of nodes V, where each node $i \in V$ is associated with a weight vectors w_i, and a set of edges between nodes. At the start of learning, the network has two nodes with weights randomly initialized. In each learning iteration, a new input stimulus ζ is observed and the following adaptation steps are performed.

1) Find the best matching node c and second best matching node c' w.r.t. ζ

$$c = \arg\min_{j \in V} \|\zeta - w_j\|_2 \tag{1}$$

$$c' = \arg\min_{j \in V/\{c\}} \|\zeta - w_j\|_2 \tag{2}$$

and add an edge between them, if it does not exist, and set its age to 0.

2) Calculate the activity a of the best matching node based on the Euclidean distance between its weight vector w_c and the input ζ

$$a = \exp\left(-\|\zeta - w_c\|_2\right). \tag{3}$$

3) If the activity a of node c is below a threshold a_T and its habituation (a measure of the node's responsiveness to input stimuli, inversely proportional to the number of times it has been a best match) is below a threshold h_T, create a new node v with a weight vector $(w_c + \zeta)/2$ and an edge to both c and c' and remove the edge between c and c'.

4) Move the weights of the best matching node c and its neighbors k, with which it shares edges, toward ζ

$$\Delta w_c = \epsilon_c \times h_c \times (\zeta - w_c) \tag{4}$$

$$\Delta w_k = \epsilon_n \times h_k \times (\zeta - w_k) \tag{5}$$

Algorithm 1 BEHAVIOR$(T_{0:n-1}) \rightarrow g^b$

Require: Growing self-Organizing networks G^{ACT}, G^{INT}, and G^B
1: Find the best matching node c^{act} in G^{ACT} w.r.t. $f_d(T_{0:n-2}; \theta^{f_d})$
2: Compute action embedding $g^{act} \leftarrow w_{c^{act}}$
3: Find the best matching node c^{int} in G^{INT} w.r.t. $f_x(T_{n-1}; \theta^{f_x})$
4: Compute intention embedding $g^{int} \leftarrow w_{c^{int}}$
5: Find the best matching node c^b in G^B w.r.t. $g^{act} \oplus g^{int}$
6: Compute behavior embedding $g^b \leftarrow w_{c^b}$
7: Return g^b

where $0 < \epsilon_n < \epsilon_c < 1$ and h_k is the habituation value for node k.

5) Decrease the habituation value for the best matching node c and its neighbors k

$$h_c = h_0 - \frac{\left(1 - e^{\frac{-\alpha_c t}{\tau_c}}\right)}{\alpha_c} \tag{6}$$

$$h_k = h_0 - \frac{\left(1 - e^{\frac{-\alpha_n t}{\tau_n}}\right)}{\alpha_n} \tag{7}$$

where h_0 is the initial habituation value. α_c,α_n and τ_c,τ_n are constants controlling the habituation curve.

6) Increment the age of all edges emanating from c by 1. If the age of any edge exceeds a threshold κ, remove that edge and remove any node with no remaining edges.

An illustration of the GWR networks G^{ACT} and G^{INT} is shown in Fig. 2(c).

Behavior Embeddings: To learn the behavior embeddings, the combined action–intention embedding space is incrementally self-organized by using a higher level GWR network G^B. During learning, the G^B follows the same adaptation steps of the standard GWR network explained earlier. At each learning iteration, the weight vectors g^{act} and g^{int} of the best matching nodes in the lower level networks G^{ACT} and G^{INT} w.r.t. an input demonstration are concatenated and fed as input to G^B, as shown is Fig. 2(c). The incremental self-organization of action–intention embeddings allows learning a growing set of behaviors, which is necessary for continual multitask learning. After the learning of the behavior embeddings, the action G^{ACT}, intention G^{INT}, and behavior G^B networks can be utilized to map a visual demonstration to the intended behavior behind the demonstration (Algorithm 1).

The two-level hierarchy of embeddings ensures that the learned behavior embeddings capture the action–intention associations and their similarities.

B. Self-Supervised Learning of Task Representations

Given the learned behavior embedding space, we aim to train a differentiable model that maps an unlabeled demonstration to a corresponding task representation. In order to do so, we transfer knowledge from the behavior self-organization explained earlier to a task inference neural network, which we call TINet. We construct a behavior-matching self-supervised learning objective to perform the knowledge transfer by training the TINet to map a given demonstration to its nearest behavior embedding in the unsupervisedly learned behavior embedding space, which we use as the target task representation.

The input to the TINet is an encoding of a demonstrated trajectory $T_{0:n-1}$ produced by the demonstration encoder f_d and the target output is the weight vector g^b of the best matching node in the behavior net G^B w.r.t. the input demonstration. The TINet model is formally described by

$$z = f_{\text{INF}}(f_d(T_{0:n-1})) \tag{8}$$

where f_{INF} is the task inference function and z is the predicted task representation. We train the TINet to minimize the following behavior-matching loss

$$\mathcal{L}_{\text{BM}} = \|f_{\text{INF}}(f_d(T_{0:n-1})) - g^b\|_2^2. \tag{9}$$

To enable task inference from incomplete demonstrations, the TINet is further trained to produce the same output for the original version $T^i_{0:n-1}$ (the complete demonstration) and the temporally cropped version $T^j_{u:v|0\le u<v\le n-1}$ (the incomplete demonstration) of an input demonstration [Fig. 2(b)] in a set of K different demonstrations, where u and v are sampled at random from $[0, n-1]$. This is performed by minimizing the following contrastive loss:

$$\mathcal{L}_C = -\log \frac{\exp\big(\text{sim}\big(z_i, z_j\big)/\tau\big)}{\sum_{k=0}^{K-1} \mathbb{1}_{[k\neq i]} \exp(\text{sim}(z_i, z_k)/\tau)} \tag{10}$$

where z_i and z_j are the task representations of the complete and incomplete versions of the input demonstration, respectively, $\mathbb{1}_{[k\neq i]} \in \{0, 1\}$ is an indicator function, τ is a temperature hyperparameter, and $\text{sim}(\cdot, \cdot)$ is a similarity function. We use cosine similarity $\text{sim}(z_i, z_j) = z_i^\top z_j/(\|z_i\|_2 \, \|z_j\|_2)$ between task representations z_i and z_j [40]. This process is shown in Fig. 2(b). The two TINet blocks are the same network which produces a task representation z_i when the input is the encoded complete demonstration $f_d(T_{0:n-1})$ and a task representation z_j when the input is the encoded incomplete demonstration $f_d(T_{u:v})$. The contrastive loss in (10) enforces that the task representations for the original ("complete") and cropped ("incomplete") demonstrations z_i and z_j, respectively, are close to each other while also preventing the TINet from always producing the same vector on the output by pushing away the task representations of different demonstrations from each other. In other words, minimizing $\mathcal{L}_C$ (10) means that the complete and incomplete versions of the input demonstration will have nearly identical task representations. Consequently, the robot can now infer the correct task representation when shown only an incomplete demonstration, because the TINet is trained to produce a task representation for the incomplete demonstration that is close to the task representation for the corresponding complete demonstration.

The overall self-supervised learning loss to train a randomly initialized TINet is

$$\mathcal{L}_{\text{SSL}} = \mathcal{L}_{\text{BM}} + \mathcal{L}_C. \tag{11}$$

By jointly optimizing for behavior-matching and contrastive prediction, as shown in (11), the TINet is trained to infer the task at hand from complete or incomplete visual demonstrations without requiring predefined task labels. In addition,

distilling the growing behavior net G^B into the TINet (9) allows the TINet to continually learn to infer new tasks.

C. End-to-End Continual Multitask Learning

Given an unlabeled input demonstration, the learning agent can use the TINet to infer the task it is required to solve. To enable learning a growing set of tasks, we use the task representation z provided by the TINet together with the current environment state s as input to a multitask policy π which is trained with RL to optimize performance over tasks. This is done by finding the policy π that minimizes the following loss:

$$\mathcal{L}_{\text{RL}} = -E_{\pi}\left(\sum_{t=0}^{\infty}\gamma^t r_t\right) \tag{12}$$

where t is the time step, r is the reward, and $\gamma \in [0, 1)$ is a discount factor. Minimizing $\mathcal{L}_{RL}$ corresponds to the standard RL objective of maximizing the expected cumulative discounted reward. The RL algorithm used to train the multitask policy in the presented work is deep deterministic policy gradient (DDPG) [41], but our approach can be paired with any other RL algorithm with minimal changes. DDPG updates the policy by gradient ascent on the action-value (Q-)function

$$\theta^{\pi} = \theta^{\pi} + \mu\nabla_a Q\Big(s, a; \theta^{Q}\Big)|_{a=\pi(s_i)}\nabla_{\theta^{\pi}}\pi\left(s; \theta^{\pi}\right) \tag{13}$$

where $Q(s, a)$ is the expected value of taking action a in state s and following policy π thereafter, θ^{π} and θ^{Q} are the policy and Q-function parameters, and μ is the gradient step size. In our implementation, the state feature vector $x = f_x(s; \theta^{f_x})$ and the task representation z are used instead of s as input to Q and π.

The whole learning architecture (Fig. 2), including the TINet TINet, is trained end-to-end. This encourages the task representations from the TINet to capture the task structure. Gradients from $\mathcal{L}_{RL}$ and $\mathcal{L}_{SSL}$ are backpropagated through the TINet, the demonstration encoder f_d, and the CNN state encoder f_x, optimizing all the networks end-to-end, as shown in Fig. 2(b). Hence, the final loss for training the multitask learning agent with our approach is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{SSL}} + \mathcal{L}_{\text{RL}}. \tag{14}$$

The complete Self-supervised task representation learning (SSTRL) algorithm for continual multitask RL is given in Algorithm 2. At the beginning of each learning episode, a visual demonstration in the form of a trajectory of frames $T_{0:n-1}$ is presented to the robot (line 5). We then compute the behavior embedding g^b (line 6) of the best matching node in the growing behavior net G^B w.r.t. $T_{0:n-1}$, as shown in Algorithm 1, followed by adjusting the networks G^{ACT}, G^{INT}, and G^B [(1)–(7)]. Random temporal cropping is performed on $T_{0:n-1}$ to generate a corresponding incomplete demonstration $T_{u:v}$ (line 8). The indices u and v are uniformly sampled from $[0, n-1]$ such that $0 \leq u < v \leq n-1$. We add the complete $T_{0:n-1}$ and incomplete $T_{u:v}$ demonstrations along with the corresponding behavior embedding g^b to the training data set $\mathcal{D}_{\text{SSL}}$ used for minimizing $\mathcal{L}_{SSL}$ (line 9). The encoded demonstration $f_d(T_{0:n-1}; \theta^{f_d})$ is passed to the TINet to infer the task representation $z = f_{\text{INF}}(f_d(T_{0:n-1}; \theta^{f_d}); \theta^{f_{\text{INF}}})$ (line 10).

Algorithm 2 SSTRL Algorithm for Continual Multitask RL

Require: An RL algorithm $\mathbb{A}$
Require: State encoder f_x, demonstration encoder f_d, task representation encoder f_{INF}, and components of $\mathbb{A}$
1: Initialize growing self-organizing networks: G^{ACT}, G^{INT}, G^B
2: Initialize datasets $\mathcal{D}_{SSL}, \mathcal{D}_{RL} \leftarrow \emptyset$
3: Randomly initialize network parameters: $\theta^{f_x}, \theta^{f_d}, \theta^{f_{INF}}, \theta^{\mathbb{A}}$
4: **for** *episode* $= 1, E$ **do**
5: Sample demonstration $T_{0:n-1}$
6: Compute $g^b = \text{BEHAVIOR}(T_{0:n-1})$ using Algorithm 1
7: Adjust G^{ACT}, G^{INT}, and G^B networks using Eq. (1)–(7) (refer to Section III-A)
8: $T_{u:v} \leftarrow \text{TEMPORALCROP}(T_{0:n-1})$
9: Insert $(T_{0:n-1}, T_{u:v}, g^b)$ into $\mathcal{D}_{SSL}$
10: $z \leftarrow f_{INF}(f_d(T_{0:n-1}; \theta^{f_d}); \theta^{f_{INF}})$
11: Sample initial state s
12: **while** not terminal **do**
13: $x \leftarrow f_x(s; \theta^{f_x})$
14: Sample action $a \sim \pi(x, z)$ using $\mathbb{A}$'s behavioral policy
15: Execute a and observe r and s'
16: Insert $(s, T_{0:n-1}, a, r, s')$ into $\mathcal{D}_{RL}$
17: Update $\theta^{f_x}, \theta^{f_d}, \theta^{f_{INF}}$ using $\mathcal{D}_{SSL}$ and $\mathcal{L}_{SSL}$ in Eq. (11)
18: Update $\theta^{f_x}, \theta^{f_d}, \theta^{f_{INF}}, \theta^{\mathbb{A}}$ with $\mathbb{A}$ using $\mathcal{D}_{RL}$ and $\mathcal{L}_{RL}$ in Eq. (12)
19: $s \leftarrow s'$
20: **end while**
21: **end for**
22: Return optimized policy π

Actions are generated by the behavioral policy π of the chosen RL algorithm $\mathbb{A}$ (line 14). The policy takes as input the state feature vector $x = f_x(s; \theta^{f_x})$ and the inferred task representation z and is parameterized by parameters θ^{π}. $\mathbb{A}$ can be a policy-gradient algorithm, in which case $\theta^{\mathbb{A}} = \{\theta^{\pi}\}$, or a value-based algorithm, in which case $\theta^{\mathbb{A}} = \{\theta^{\pi}, \theta^{Q}\}$, where Q is the action-value function. In our implementation, the RL algorithm $\mathbb{A}$ used is DDPG [41] which is value based and updates the policy by gradient ascent on the Q-function (13) using experiences sampled from $\mathcal{D}_{RL}$ (line 18). The learning parameters of our task inference architecture are updated to minimize the total loss $\mathcal{L}_{\text{total}}$ (14).

IV. Experimental Results

In this section, we empirically evaluate our proposed SSTRL algorithm for continual multitask RL on learning a fixed as well as a growing set of visuomotor tasks with a humanoid robot and compare it against three multitask learning baselines: 1) Plan2Explore [30]; 2) Progress&Compress (P&C) [34]; and 3) behavior-guided policy optimization (BGPO) [16]. Additionally, we perform ablation experiments to investigate the effect of the different components of our approach on its performance. We also perform one-shot generalization experiments to test the performance of our approach on a set of unseen tasks based on a single visual demonstration.

A. Experimental Setup

Hyperparameters: The CNN state encoder has three 3×3 convolutions with 32, 64, and 128 channels, respectively. Each convolution is followed by ReLU activation and 2×2 max-pooling. This is followed by two fully connected layers each with 128 units and ReLU activations. The demonstration

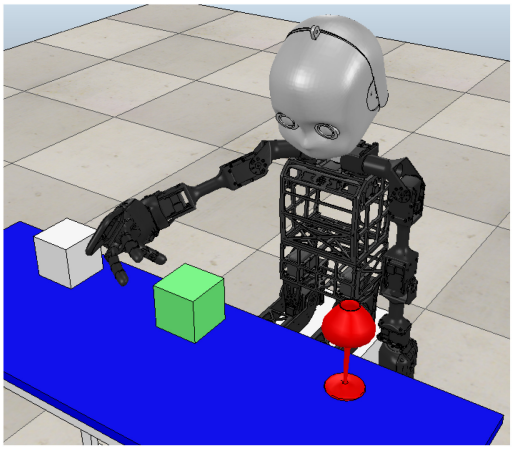

Fig. 3. NICO robot in the simulation environment facing a table with three objects.

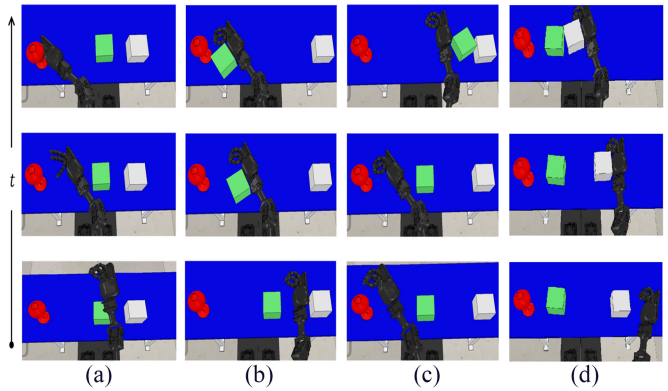

Fig. 4. First-person demonstrations of four visuomotor tasks: (a) "Grasp the red glass," (b) "Push the green box toward the red glass," (c) "Push the green box toward the white box," (d) "Push the white box toward the green box." From bottom to top: RGB frames of initial, intermediate, and terminal configurations.

encoder uses a single-layer LSTM with 256 hidden units and tanh activations. The TINet is a fully connected multilayer perceptron (MLP) of three layers with 512, 512, and 256 units, respectively. The multitask policy and Q-functions are parameterized by a 2-layer MLP each. The hidden layer is 64-D with ReLU activation. The output layer contains a single unit with linear activation in the Q-network and d units with tanh activations in the policy network, where d denotes the dimenstionality of the action space. The training data sets $\mathcal{D}_{\text{SSL}}$ and $\mathcal{D}_{\text{RL}}$ are stored in memory buffers of sizes 10^5 and 10^6, respectively. All networks are trained using the Adam optimizer [42] with learning rate 0.001 and batch size 256. The discount factor γ is set to 0.99. We do not use any hyperparameter for balancing the behavior-matching loss and the contrastive loss to simplify the training process. The details on the hyperparameters of the growing Action Net G^{ACT}, Intention Net G^{INT}, and Behavior Net G^{B} are given in Appendix A. Training is done with Tensorflow [43] on a desktop with Intel i5-6500 CPU and a single NVIDIA Geforce GTX 1050 Ti GPU.

Robotic Setup: We conduct our experiments on the neuro-inspired companion (NICO) robot [44] using the CoppeliaSim (formerly V-REP) robot simulator [45]. Real-world experiments are described in Section IV-F. Fig. 3 shows the simulated NICO sitting in front of a table on top of which different objects are placed. In all experiments, we consider a motor action controlling four degrees of freedom in the right arm: two joints in the shoulder, one joint in the elbow, and one joint in the hand. The shoulder and elbow joints have an angular range of motion of $\pm 100°$ and $\pm 85°$, respectively. The tendon-operated multifingered hand consists of one thumb and two index fingers with finger joints having an angular range of motion of 0°–160°. The input to the robot learning algorithm is a 64×32 RGB image obtained from a vision sensor. Examples of the original output of the vision sensor are shown in Fig. 4.

B. Multitask Learning Evaluation

In our experiments, we consider the following visuomotor tasks: Grasp the red glass (*Task-1*), push the green box toward the red glass (*Task-2*), push the green box toward the white box (*Task-3*), and push the white box toward the green box (*Task-4*). We collect 1000 visual demonstrations per task with random initial robot configuration and object positions (see Fig. 4). The demonstrations have an average length of 30 steps ($\approx$ 6 s). Due to the demonstrations having variable lengths, we apply zero-padding and masking to them when training the LSTM network of the demonstration encoder. During learning, a task is randomly sampled at the start of each episode which terminates when the task is successfully completed or after a maximum of 50 timesteps. A reward of 1 is given for a successful task completion and 0 otherwise. At the end of each episode, the learned policy is tested by randomly sampling a task and running the policy for 50 timesteps.

All learning algorithms use the environment state as input to the policy. Plan2Explore and P&C assume that changes in task identity are known to the agent, while BGPO and SSTRL perform task inference from unlabeled input demonstrations and use the inferred task representation as additional input to the policy. For implementing Plan2Explore, we train a global world model from task-agnostic experience gathered during learning by an exploration policy trained to maximize the expected novelty over future model states. In each test episode, the policy for the sampled task is trained in imagination using the model and then executed for 50 timesteps. P&C has two policy networks with layerwise connections between them: the active column and the knowledge base. At each learning episode, the active column is trained on a sampled task and then distilled into the knowledge base which is then evaluated. BGPO and SSTRL receive an unlabeled demonstration from a randomly sampled task at the start of each episode and use it to infer the task representation. The policy conditioned on the inferred representation and environment state is trained for one episode and evaluated on a random task. The implementation details of the baseline algorithms can be found in Appendix B. We perform our experiments in two settings: 1) fixed-set multitask learning, where the set of tasks the robot is required to learn is kept fixed throughout the course of learning and 2) continual multitask learning, where the tasks are presented sequentially to the robot.

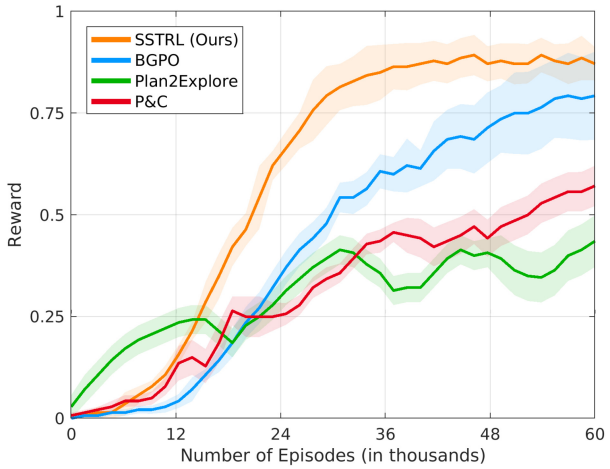

Fig. 5. Performance curves of P&C, Plan2Explore, BGPO, and SSTRL on learning a fixed set of four independent visuomotor control tasks with the NICO robot. Shaded regions represent one standard deviation over ten random seeds.

Fig. 5 shows the total reward per test episode for each algorithm in the fixed-set multitask learning setting, averaged over ten random seeds. As shown in the figure, Plan2Explore performs slightly better than the other algorithms over the first 12K episodes. However, the performance tends to be largely unstable thereafter, with the average reward staying under 0.5 (i.e., below 50% success rate). This is likely the result of the world model being trained on parts of the environment that are not relevant to the task at test time. In contrast, the performance of P&C continues to improve with more training but reaches only an average reward of 0.57 by the end of learning. Since the knowledge base parameters in P&C are restricted to be close to their previously trained values when the policy learned by the active column is distilled into the knowledge base, the multitask policy of the knowledge base can only generalize slowly. Consequently, its performance on a given task heavily depends on how similar that task is to the recently learned one, which may explain the slow increase in the observed average reward over time. Instead of mitigating interference among tasks by regularizing the update to the multitask policy parameters, which still leads to interference since tasks are learned in a joint parameter space, BGPO and SSTRL avoid interference in the first place by conditioning the policy on a task representation which is learned in a space different than that of the policy parameters. While BGPO and SSTRL show a better final performance, achieving an average reward of over 0.75 at the end of learning, SSTRL has a more stable performance and faster convergence than BGPO.

We also evaluate the performance of the trained policy of each algorithm on the individual tasks. This includes a comparison to a single-task policy optimization, where a separate policy network is trained with DDPG [41] on each task individually (see Appendix B for implementation details). The trained policy attempts each task 100 times. The success rate is given in Table I, with SSTRL achieving the highest success rate in three out of four tasks. The results suggest that multitask learning with SSTRL not only allows the policy to accomplish a number of different tasks but also to improve its performance on each individual task via sharing policy and task representations.

TABLE I
PERFORMANCE COMPARISON OF THE ALGORITHMS ON INDIVIDUAL TASKS. THE REPORTED NUMBERS ARE SUCCESS RATES OVER 100 TRIALS

Algorithm	Task-1	Task-2	Task-3	Task-4
Single-task policy optimization	53%	72%	66%	60%
P&C [34]	23%	55%	31%	18%
Plan2Explore [30]	35%	43%	29%	25%
BGPO [16]	54%	82%	68%	**71%**
SSTRL (Ours)	**67%**	**86%**	**79%**	68%

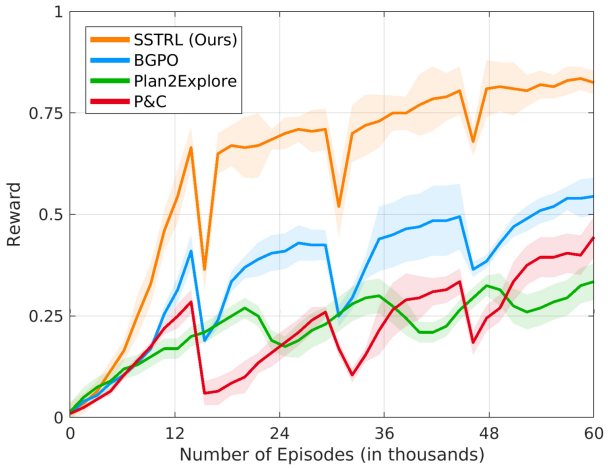

Fig. 6. Performance curves of P&C, Plan2Explore, BGPO, and SSTRL in the continual multitask learning setting. Shaded regions represent one standard deviation over ten random seeds.

In the continual learning setting, the learning starts with *Task-1*. *Task-2*, *Task-3*, and *Task-4* are presented after 15K, 30K, and 45K episodes, respectively. At test time, each algorithm is evaluated on a task randomly sampled from the presented tasks. We plot the total reward per test episode, averaged over ten random seeds, in Fig. 6. A drop in performance can be observed for all algorithms after each new task is introduced. Changing tasks has less direct effect on Plan2Explore as it follows task-agnostic exploration policy during learning. However, the policy learned offline at test time relies on a world model that may have been trained on task-irrelevant data. Thus, the policy performs poorly on tasks for which the model is not sufficiently trained, particularly when such tasks are visited for several episodes before new tasks are presented, as it is the case in the continual learning setting. Similarly, when P&C encounters an unseen task, the knowledge base's policy typically requires longer training before it adapts to that task, because the active column's policy, which is distilled into the knowledge base after every episode, will likely fail to quickly solve the unseen task. This leads to a considerably small improvement in performance during the intervals between the tasks, as shown in Fig. 6.

BGPO and SSTRL, on the other hand, are originally more robust to learning instability caused by the sequential presentation of tasks due to their self-organizing networks that grow when they cannot find a behavior embedding that closely

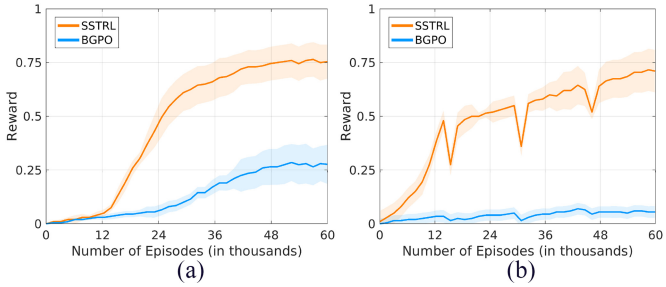

Fig. 7. Performance curves of BGPO and SSTRL on incomplete demonstrations in two multitask learning settings: (a) fixed-set and (b) continual. Shaded regions represent one standard deviation over ten random seeds.

matches an input demonstration. By using a task representation based on a growing behavior network as input to the policy, they can generalize faster to new tasks and maintain the performance on old tasks while learning new ones since different tasks have different representations on which the policy is conditioned. Compared to BGPO that reached an average reward of only 0.55 after 60K episodes, SSTRL was able to converge to an average reward of over 0.8. This empirically shows the advantage of learning separate action and intention embeddings and the advantage of the TINet that learns a generalizable mapping from demonstrations to task representations end-to-end, which is particularly useful in the continual learning setting.

C. Performance Evaluation on Incomplete Demonstrations

In this experiment, we aim to compare the performance of the multitask policy trained with BGPO and SSTRL when incomplete demonstrations are used as input. We make two comparisons in the fixed-set and continual learning settings. At each test episode, we randomly sample ten unlabeled demonstrations and apply random temporal cropping to each demonstration. We then run the trained policy ten times each using a different temporally cropped demonstration as input. The results are shown in Fig. 7. In the fixed-set setting, BGPO reaches an average reward of around 0.25, while SSTRL appears to achieve three times more average reward by the end of learning. The difference in performance between the two algorithms is more pronounced in the continual learning setting. As opposed to BGPO which fails to make any progress in maximizing the obtained reward over the entire learning process, SSTRL is able to continue to improve its performance after every new task is presented. This demonstrates the effectiveness of training the TINet to learn a joint representation of original and cropped versions of unlabeled demonstrations in SSTRL, enabling task inference from incomplete demonstrations and improving the performance of the multitask policy on tasks inferred form such demonstrations.

D. Ablation Study

We perform an ablation study to investigate the contribution of each component of our proposed approach to the overall performance in the continual learning setting and plot the results in Fig. 8. When the action and intention networks are removed ("*no-*$G^{\text{ACT,INT}}$"), the behavior network

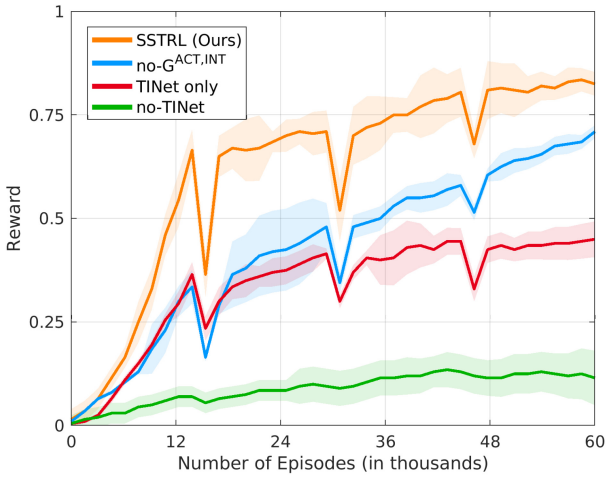

Fig. 8. Performance curves of SSTRL with all of its components and after removing different components in the continual multitask learning setting. Shaded regions represent one standard deviation over ten random seeds.

TABLE II
SUCCESS RATE PER TASK FOR EACH ABLATION CONFIGURATION OVER THE TEST EPISODES

Algorithm	Task-1	Task-2	Task-3	Task-4
no-$G^{ACT,INT}$	65.33%	61.12%	45.80%	33.00%
TINet only	42.35%	37.19%	30.51%	17.79%
no-TINet	11.98%	10.33%	5.56%	4.95%
SSTRL (Ours)	**74.32%**	**71.00%**	**68.76%**	**56.20%**

G^B is forced to learn the behavior embeddings directly from demonstrations. This slows convergence as the robot lacks the information the action–intention associations offer to facilitate task inference. Removing the hierarchical self-organization architecture altogether and keeping the TINet ("*TINet only*") causes the multitask policy to converge to a lower average reward than when self-organization of behaviors is enabled. Since, in this case, the TINet cannot be trained to map demonstrations to best matching behaviors, the algorithm will likely fail to infer the intended behavior behind each demonstration, thus preventing the policy from achieving high performance across all tasks. If the TINet is removed ("*no-TINet*"), the policy exhibits poor performance, with the average reward staying under 0.25 until the end of learning. The success rate per task for each of the considered Configurations is given in Table II.

These results suggest that behavior self-organization is essential for task inference in SSTRL and that, without the behavior-matching loss, the TINet can only reach an average performance, indicating that training the TINet with the contrastive loss is sufficient for improving the performance compared to the case when the TINet is removed. Additionally, unsupervised learning of action and intention embeddings with G^{ACT} and G^{INT} networks and using the combined action–intention embeddings as input to the G^B network allow behavior self-organization to significantly improve learning speed and final performance compared to directly using the encoded demonstrations as input to G^B.

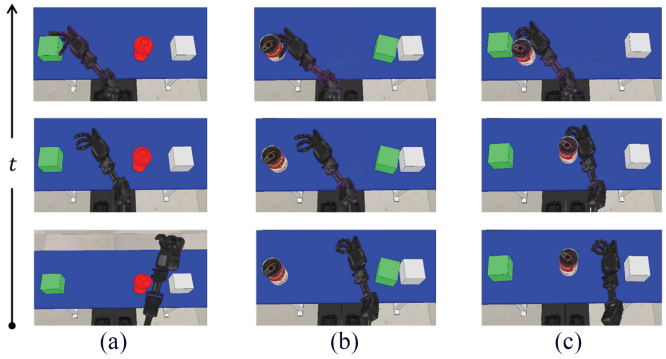

Fig. 9. Example demonstrations of three held-out visuomotor tasks: (a) "Grasp the green box," (b) "Grasp the can," (c) "Push the can toward the green box." From bottom to top: RGB frames of initial, intermediate, and terminal configurations.

TABLE III
ONE-SHOT GENERALIZATION PERFORMANCE ON HELD-OUT TASKS. THE REPORTED NUMBERS ARE SUCCESS RATES OVER 100 TRIALS

Algorithm	Task-5	Task-6	Task-7
BGPO [16]	22%	0%	0%
SSTRL (Ours)	**55%**	**35%**	**61%**

E. One-Shot Task Generalization

We evaluate the multitask policy trained in the continual learning setting with BGPO and SSTRL (the best-performing policy out of ten training runs) on a held-out set of three unseen tasks: Grasp the green box (*Task-5*), grasp the can (*Task-6*), and push the can toward the green box (*Task-7*). For each unseen task, we provide the trained policy with one visual demonstration of successful task execution as input. Fig. 9 shows an example demonstration for each task. We perform 100 test trials with 100 different demonstrations per unseen task. The success rate for the unseen tasks is given in Table III.

The held-out tasks are chosen such that they include combinations of action–object pairs (*Task-5*, *Task-6*) and object-object pairs (*Task-7*) that were not seen during training. As shown in Table III, SSTRL achieves a nonzero success rate in all the held-out tasks. We find that *Task-6* is particularly challenging. We believe this is because it not only involves a novel object (the can) but also because the policy has learned to *grasp* only a single object (the red glass) during training whereas it has learned to *push* different objects. Nevertheless, the multitask policy trained with SSTRL appears to capture the intention of the observed demonstration despite having never seen any demonstration of the related task. We demonstrate the one-shot generalization performance of our approach on the held-out tasks in the accompanying video (https://youtu.be/77cP8ciHcII). Analyzing the trajectories generated by the policy, we observe that while the robot does not exactly complete the task in the unsuccessful trials, it moves toward the target object, and in many cases the robot does come fairly close to the target object but fails only to close the fingers correctly on the object or to place the pushed object right next to the target object. This clearly indicates that the inferred task representation is informative of the task at hand.

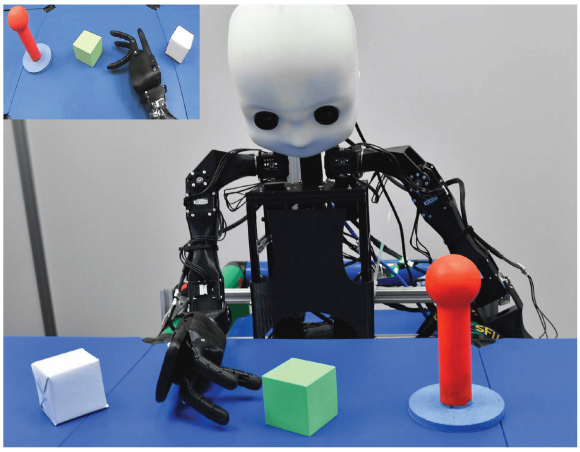

Fig. 10. Exocentric and egocentric (inset) views of the real-world experimental setup for multitask learning from the perspective of the NICO robot.

TABLE IV
PERFORMANCE EVALUATION IN THE REAL WORLD. THE REPORTED NUMBERS ARE SUCCESS RATES OVER 20 TEST EPISODES

Algorithm	Task-1	Task-2	Task-3	Task-4
P&C [34]	15%	25%	15%	20%
Plan2Explore [30]	10%	30%	20%	25%
BGPO [16]	35%	**75%**	50%	50%
SSTRL (Ours)	**60%**	**75%**	**65%**	**70%**

F. Performance Evaluation in the Real World

We compare the performance of the multitask policy trained under the continual learning setting in simulation on the real NICO robot. For each algorithm, we take the best-performing policy out of 10 simulation-based training runs and perform 20 test episodes per task on the real robot, each with different object positions and initial robot configuration. For BGPO and SSTRL, we use a random simulation-based demonstration as input to the policy.

The simulation uses a URDF model of the NICO robot and, thus, there is no difference between the real NICO and the simulated NICO. The height and color of the table and of the robot's seat are identical in the simulation and the real environment, which facilitates a direct transfer of the robot's arm pose and the trained policy. The objects on the table are slightly different in geometry to enable more stable manipulation but have the same colors as in simulation. Fig. 10 shows the experimental setup with the real NICO robot. We do not perform any additional training or fine-tuning of the learning architecture and deploy it directly on the real NICO. We report the success rate per task for each algorithm in Table IV. Example runs of the multitask policy learned using SSTRL on the real robot are shown in the accompanying video (https://youtu.be/77cP8ciHcII).

V. CONCLUSION

In this article, we presented a novel self-supervised task inference approach for continual robot learning. Our approach uses a two-level hierarchical self-organization architecture to learn task-descriptive behavior embeddings from unlabeled

vision-based demonstrations. In the lower level, action and intention embeddings are incrementally learned from self-organization of the observed movement and effect parts of each demonstration. The self-organization of the combined action–intention embeddings constructs a higher level behavior embedding space. A novel behavior-matching self-supervised learning objective is used to train a TINet, which we call TINet, to map complete and incomplete versions of an unlabeled demonstration to a best matching behavior in the learned behavior embedding space, which we use as the target task representation. A multitask policy is built on top of the TINet and trained with RL to optimize performance over tasks. The whole learning architecture, including the TINet, is trained end-to-end, encouraging the inferred task representation to capture the structure of the task at hand. We evaluate our approach in the fixed-set and continual multitask learning settings and compare it to different multitask learning baselines. The results show that our approach outperforms the other baselines, with the difference being more pronounced in the challenging continual learning setting. Our approach also achieves higher task performance than the compared demonstration-based baseline on incomplete input demonstrations. Additionally, the results from one-shot task generalization experiments clearly demonstrate the ability of our approach to read the intention behind a task demonstration and generate a meaningful action trajectory to complete the task without any learning on the task.

In contrast to previous multitask learning approaches, our approach is constant in the number of policy parameters, makes no assumptions about task distribution, and while maintaining performance on previously learned tasks, avoids learning interference among tasks as it accelerates learning progress on new tasks. The ability to infer the intended behavior behind a visual demonstration rather than copying and memorizing the observed actions allows our approach to complete the tasks more efficiently than the provided demonstrations, especially when the input demonstration is imperfect or incomplete. It is also worth mentioning that the policy trained with our approach performs the desired task even when the initial environment state, including robot configuration, is different between the demonstration and test-time settings, as shown in the accompanying video (https://youtu.be/77cP8ciHcII).

One limitation of our approach is that it requires additional computational time for constructing and adapting the growing self-organizing networks necessary for task inference. However, this happens only once every learning episode and never at test time, where only the TINet is queried for a task representation without any search in the graph of the growing networks. Besides, this computational complexity scales only linearly with the number of demonstrations. In its current form, our approach uses first-person demonstrations. One exciting direction for future work is to adapt our approach to address task inference when the morphology of the demonstrator and the robot are different, which is the case when using third-person demonstrations from a human teacher. Another direction for future work is to train the approach on multimodal demonstrations using vision-and-language task descriptions.

TABLE V
HYPERPARAMETERS OF G^{ACT} NET USED IN OUR EXPERIMENTS

Hyperparameter	Value
Activity Threshold	$a_T = 0.7$
Habituation Threshold	$h_T = 0.2$
Learning Rates	$\epsilon_c = 0.1, \epsilon_n = 0.05$
Initial Habituation	$h_0 = 1$
Habituation Curve	$\alpha_c = \alpha_n = 1.05, \tau_c = 0.5, \tau_n = 2$
Maximum Age Threshold	$\kappa = 80$

TABLE VI
HYPERPARAMETERS OF G^{INT} NET USED IN OUR EXPERIMENTS

Hyperparameter	Value
Activity Threshold	$a_T = 0.9$
Habituation Threshold	$h_T = 0.3$
Learning Rates	$\epsilon_c = 0.1, \epsilon_n = 0.01$
Initial Habituation	$h_0 = 1$
Habituation Curve	$\alpha_c = \alpha_n = 1.05, \tau_c = 1, \tau_n = 2.7$
Maximum Age Threshold	$\kappa = 100$

TABLE VII
HYPERPARAMETERS OF G^{B} NET USED IN OUR EXPERIMENTS

Hyperparameter	Value
Activity Threshold	$a_T = 0.8$
Habituation Threshold	$h_T = 0.15$
Learning Rates	$\epsilon_c = 0.1, \epsilon_n = 0.01$
Initial Habituation	$h_0 = 1$
Habituation Curve	$\alpha_c = \alpha_n = 1.05, \tau_c = 3.3, \tau_n = 14.3$
Maximum Age Threshold	$\kappa = 90$

APPENDIX A
HYPERPARAMTERS OF THE GROWING SELF-ORGANIZING NETWORKS

Here, we give details on the hyperparameters of the growing Action Net G^{ACT}, Intention Net G^{INT}, and Behavior Net G^{B} in Tables V–VII, respectively.

APPENDIX B
BASELINE DETAILS

Plan2Explore: The convolutional image encoder and decoder networks are the same from [46]. We use an ensemble of five transition models whose prediction disagreement is used to derive the intrinsic reward for the task-agnostic exploration, with each model implemented as a two hidden-layer MLP which takes the recurrent state of the recurrent state space model (RSSM) [47] and the action as input and predicts 1024-D image encoder features. The reward prediction, state-value, and policy functions are parameterized by a three hidden-layer MLP each, with 200 ReLU units in each layer, and trained using Adam [42] with a learning rate of 10^{-5}. The imagination horizon is 15.

P&C: The policy and value function of the active column and knowledge base share a convolutional encoder of two Conv layers with 16 6×6 and 32 3×3 filters and ReLU activations, followed by a fully connected layer with 256 units and ReLU activations. A separate, fully connected output layer with linear activation is used for each of the policy and value function. The networks are trained using Adam [42] and with

a learning rate of 0.003. The forgetting coefficient and Fischer regularization strength are set to 0.95 and 25, respectively.

BGPO: A variational autoencoder (VAE) [48] is used to learn the state representation. The VAE encoder consists of three Conv layers with 32, 64, and 128 3×3 filters, each followed by ReLU activation and 2×2 max-pooling, and two fully connected layers of 128 linear units outputting the mean and standard deviation of a diagonal Gaussian from which state representations are sampled. The decoder mirrors the encoder, but uses a sigmoid activation in the output layer. The inverse model used is a 1-layer MLP with tanh activation. Each demonstration is a sequence of VAE-encoded states. An LSTM autoencoder with 64 hidden units in the encoder and decoder LSTMs is used to encode the demonstrations. We use the same hyperparameters that were used in [16] for the GWR-B model. DDPG is used as the base RL algorithm. The policy and action-value function are parameterized by an MLP with a hidden layer of 64 ReLU units and output layer of one linear unit in the action-value function and 4 tanh units in the policy. The VAE, inverse model and LSTM autoencoder are pretrained on 4000 task demonstrations and then fixed during policy learning. The networks are trained using Adam [42] with learning rate 0.001.

Single-Task Policy Optimization: In both the single-task Q- and policy networks, a CNN state encoder (described in Section IV-A) is used, followed by one fully connected hidden layer with 64 ReLU units. The output layer in the policy network is a tanh layer and in the Q-network is a linear layer. The actions are included after the CNN encoder in the Q-network. The networks are trained with DDPG [41] using Adam [42] with a learning rate of 0.001 and a batch size of 256.

ACKNOWLEDGMENT

The authors thank Erik Strahl for his technical support with the real-world experimental setup.

REFERENCES

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.

[2] I. Akkaya et al., "Solving rubik's cube with a robot hand," 2019, *arXiv:1910.07113*.

[3] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," in *Proc. Robot. Sci. Syst. (RSS)*, Jun. 2019, pp. 1–10.

[4] Y. Zhu et al., "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017, pp. 3357–3364.

[5] A. A. Rusu et al., "Policy distillation," in *Proc. ICLR*, 2016, pp. 1–13.

[6] Y. W. Teh et al., "Distral: Robust multitask reinforcement learning," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 4499–4509.

[7] O. Watkins, A. Gupta, T. Darrell, P. Abbeel, and J. Andreas, "Teachable reinforcement learning via advice distillation," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2021, pp. 6920–6933.

[8] R. Mendonca, A. Gupta, R. Kralev, P. Abbeel, S. Levine, and C. Finn, "Guided meta-policy search," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 9653–9664.

[9] A. Zhou et al., "Watch, try, learn: Meta-learning from demonstrations and reward," in *Proc. Int. Conf. Learning Represent. (ICLR)*, 2020, pp. 1–13.

[10] C. Lynch and P. Sermanet, "Language conditioned imitation learning over unstructured data," in *Proc. Robot. Sci. Syst. (RSS)*, 2021.

[11] A. Silva, N. Moorman, W. Silva, Z. Zaidi, N. Gopalan, and M. Gombolay, "LanCon-learn: Learning with language to enable generalization in multi-task manipulation," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1635–1642, Apr. 2022.

[12] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 5331–5340.

[13] J. Zhang et al., "MetaCURE: Meta reinforcement learning with empowerment-driven exploration," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 12600–12610.

[14] R. A. Williamson and E. M. Markman, "Precision of imitation as a function of preschoolers' understanding of the goal of the demonstration," *Develop. Psychol.*, vol. 42, no. 4, pp. 723–731, 2006.

[15] G. Rizzolatti and C. Sinigaglia, "The functional role of the parieto-frontal mirror circuit: interpretations and misinterpretations," *Nat. Rev. Neurosci.*, vol. 11, no. 4, pp. 264–274, 2010.

[16] M. B. Hafez and S. Wermter, "Behavior self-organization supports task inference for continual robot learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2021, pp. 6739–6746.

[17] E. Jang et al., "BC-Z: Zero-shot task generalization with robotic imitation learning," in *Proc. Conf. Robot Learn. (CoRL)*, 2022, pp. 991–1002.

[18] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 3758–3765.

[19] C. Lynch et al., "Learning latent plans from play," in *Proc. Conf. Robot Learn. (CoRL)*, 2020, pp. 1113–1132.

[20] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, "Imitation from observation: Learning to imitate behaviors from raw video via context translation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 1118–1125.

[21] D. Kalashnikov et al., "MT-Opt: Continuous multi-task robotic reinforcement learning at scale," 2021, *arXiv:2104.08212*.

[22] A. N. Meltzoff, "Understanding the intentions of others: re-enactment of intended acts by 18-month-old children," *Develop. Psychol.*, vol. 31, no. 5, pp. 838–850, 1995.

[23] M. Tomasello, M. Carpenter, J. Call, T. Behne, and H. Moll, "Understanding and sharing intentions: The origins of cultural cognition," *Behav. Brain Sci.*, vol. 28, no. 5, pp. 675–691, 2005.

[24] C.-T. Huang, C. Heyes, and T. Charman, "Infants' behavioral reenactment of "failed attempts": Exploring the roles of emulation learning, stimulus enhancement, and understanding of intentions," *Develop. Psychol.*, vol. 38, no. 5, pp. 840–855, 2002.

[25] B. Elsner, "Infants' imitation of goal-directed actions: The role of movements and action effects," *Acta psychologica*, vol. 124, no. 1, pp. 44–59, 2007.

[26] M. Schönebeck and B. Elsner, "ERPs reveal perceptual and conceptual processing in 14-month-olds' observation of complete and incomplete action end-states," *Neuropsychologia*, vol. 126, pp. 102–112, Mar. 2019.

[27] M. Nielsen, "12-month-olds produce others' intended but unfulfilled acts," *Infancy*, vol. 14, no. 3, pp. 377–389, 2009.

[28] M. Paulus, S. Hunnius, and H. Bekkering, "Neurocognitive mechanisms underlying social learning in infancy: Infants' neural processing of the effects of others' actions," *Soc. Cogn. Affect. Neurosci.*, vol. 8, no. 7, pp. 774–779, 2013.

[29] R. P. Cooper, R. Cook, A. Dickinson, and C. M. Heyes, "Associative (not hebbian) learning and the mirror neuron system," *Neurosci. Lett.*, vol. 540, pp. 28–36, Apr. 2013.

[30] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak, "Planning to explore via self-supervised world models," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 8583–8592.

[31] A. Sharma, M. Ahn, S. Levine, V. Kumar, K. Hausman, and S. Gu, "Emergent real-world robotic skills via unsupervised off-policy reinforcement learning," in *Proc. Robot. Sci. Syst. (RSS)*, Jul. 2020, pp. 1–10.

[32] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim, "Demonstration-guided reinforcement learning with learned skills," in *Proc. Conf. Robot Learn. (CoRL)*, 2022, pp. 729–739.

[33] S. Sodhani, A. Zhang, and J. Pineau, "Multi-task reinforcement learning with context-based representations," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 9767–9779.

[34] J. Schwarz et al., "Progress & compress: A scalable framework for continual learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 4528–4537.

[35] M. Hessel, H. Soyer, L. Espeholt, W. Czarnecki, S. Schmitt, and H. van Hasselt, "Multi-task deep reinforcement learning with popart," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 3796–3803.
[36] J. Oh, S. Singh, H. Lee, and P. Kohli, "Zero-shot task generalization with multi-task deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 2661–2670.
[37] A. K. Lampinen and J. L. McClelland, "Transforming task representations to perform novel tasks," *Proc. Nat. Acad. Sci.*, vol. 117, no. 52, pp. 32970–32981, 2020.
[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[39] S. Marsland, J. Shapiro, and U. Nehmzow, "A self-organising network that grows when required," *Neural Netw.*, vol. 15, nos. 8–9, pp. 1041–1058, 2002.
[40] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 1597–1607.
[41] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, p. 13.
[43] M. Abadi et al., "Tensorflow: A system for large-scale machine learning," in *Proc. OSDI*, 2016, pp. 265–283.
[44] M. Kerzel, E. Strahl, S. Magg, N. Navarro-Guerrero, S. Heinrich, and S. Wermter, "NICO–neuro-inspired companion: A developmental humanoid robot platform for multimodal interaction," in *Proc. IEEE Int. Symp. Robot Human Interact. Commun. (RO-MAN)*, 2017, pp. 113–120.
[45] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly V-REP): A versatile and scalable robot simulation framework," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2013, pp. 1321–1326. [Online]. Available: www.coppeliarobotics.com
[46] D. Ha and J. Schmidhuber, "World models," 2018, *arXiv:1803.10122*.
[47] D. Hafner et al., "Learning latent dynamics for planning from pixels," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 2555–2565.
[48] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–14.

Muhammad Burhan Hafez received the M.Sc. degree in computer science from the University of Malaya, Kuala Lumpur, Malaysia, in 2015, and the Ph.D. degree in computer science from the University of Hamburg, Hamburg, Germany, in 2020.

He is currently a Postdoctoral Research Associate with the Knowledge Technology Group, University of Hamburg. His current research interests include machine learning and cognitive robotics, with a focus on robot skill learning.

Stefan Wermter (Member, IEEE) received the M.Sc. degree in computer science from the University of Massachusetts, Amherst, MA, USA, and the Ph.D. (Habilitation) degree in computer science from the University of Hamburg, Hamburg, Germany.

He is a Full Professor with the University of Hamburg, Hamburg, Germany, and the Director of the Knowledge Technology Institute, Department of Informatics. He has previously held positions at the University of Dortmund, Dortmund, Germany; University of Massachusetts Amherst, Amherst, MA, USA; the International Computer Science Institute, Berkeley, CA, USA; and the University of Sunderland, Sunderland, U.K. His main research interests are in neural networks, hybrid knowledge technology, neuroscience-inspired computing, cognitive robotics, natural language processing, and human–robot interaction.

Prof. Wermter is the Coordinator of the International Doctoral Training Network TRAIL, the Co-Coordinator of the International Collaborative Research Centre on Crossmodal Learning (TRR-169) and currently serves as the President for the European Neural Network Society. He has been an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and is on the advisory board of *Connection Science* and *International Journal for Hybrid Intelligent Systems* and the editorial board of the *Cognitive Computation*, *Neurosymbolic Artificial Intelligence*, and *Journal of Computational Intelligence*.