scientific reports

OPEN

Continual deep reinforcement learning with task-agnostic policy distillation

Muhammad Burhan Hafez[1,3] & Kerim Erekmen[2,3]

Central to the development of universal learning systems is the ability to solve multiple tasks without retraining from scratch when new data arrives. This is crucial because each task requires significant training time. Addressing the problem of continual learning necessitates various methods due to the complexity of the problem space. This problem space includes: (1) addressing catastrophic forgetting to retain previously learned tasks, (2) demonstrating positive forward transfer for faster learning, (3) ensuring scalability across numerous tasks, and (4) facilitating learning without requiring task labels, even in the absence of clear task boundaries. In this paper, the Task-Agnostic Policy Distillation (TAPD) framework is introduced. This framework alleviates problems (1)–(4) by incorporating a task-agnostic phase, where an agent explores its environment without any external goal and maximizes only its intrinsic motivation. The knowledge gained during this phase is later distilled for further exploration. Therefore, the agent acts in a self-supervised manner by systematically seeking novel states. By utilizing task-agnostic distilled knowledge, the agent can solve downstream tasks more efficiently, leading to improved sample efficiency. Our code is available at the repository: https://github.com/wabbajack1/TAPD.

Animals possess the inherent capacity for lifelong learning, continuously acquiring, refining and passing on knowledge and skills through entangled neurocognitive mechanisms. These mechanisms are critical for both the development of sensorimotor skills and the consolidation and retrieval of long-term memory[1]. In the complex and rapidly evolving field of artificial intelligence (AI), continual learning plays a critical role. This is particularly evident in deep reinforcement learning (RL), a subset of machine learning in which agents aim to maximize cumulative rewards by interacting with their environment[2]. The ability of these agents to learn, adapt, and apply knowledge from processing continuous streams of information in different tasks without corrupting previously acquired information is crucial[3,4]. This enables them to become universal problem solvers.

Implementing an exploratory strategy is critical for universal agents[4,5], as it is foundational to task-agnostic learning (TAL), a paradigm where agents learn without a specific task in mind[6]. Curiosity allows agents to discover and learn from unfamiliar situations, which is essential for effectively accomplishing future tasks. This is particularly crucial in environments with sparse rewards, where acquiring a particular skill can be challenging due to a lack of feedback[2,7–9]. In task-agnostic learning, agents do not have access to task-specific labels during training, focusing instead on learning general representations from the data that can be useful across a variety of tasks[8,10]. This form of learning is often associated with unsupervised learning, allowing models to leverage abundant unlabeled data efficiently[6]. The learned representations can later be fine-tuned for specific tasks, a process known as transfer learning, enabling the development of models that can adapt to a multitude of tasks and achieve improved performance when compared to models trained from scratch. Curiosity-driven exploration, inherent to task-agnostic learning, is key to developing agents that not only respond to their environment but also actively seek out new knowledge and experiences. It prepares them to overcome unexpected challenges and to take advantage of opportunities, promoting flexibility and generalization in learning[7].

To realize the potential of universal agents, one of the biggest challenges is memory consumption, as the accumulation of knowledge can quickly overwhelm available resources, especially when systems are deployed on devices with limited memory. Another issue is scalability, as the complexity of the learning process increases with the addition of new tasks. Studies by Kirkpatrick et al.[11], Rusu et al.[12] and Parisi et al.[4] have highlighted these challenges and suggested possible solutions. Rusu et al.[12] and Schwarz et al.[13] respectively, have proposed

[1]School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, United Kingdom. [2]Department of Informatics, University of Hamburg, Hamburg, Germany. [3]These authors contributed equally: Muhammad Burhan Hafez and Kerim Erekmen. ✉email: burhan.hafez@soton.ac.uk

the development of algorithms capable of not forgetting and managing growing parameters. Despite promising progress and potential solutions, there are still unsolved challenges in the field of continual reinforcement learning. One of these challenges is learning without clear task boundaries. It is difficult to define when one task ends and another begins, resulting in the need for learning in a task-agnostic manner[14]. Achieving optimal performance for specific tasks also remains a challenge, as universal agents must continuously adapt to non-stationary data and refine their knowledge to achieve high performance in different contexts, seamlessly transferring knowledge across tasks[7,15]. Finding innovative solutions to fully exploit the potential of continual learning in reinforcement learning is required.

In this paper, we introduce the *Task-Agnostic Policy Distillation* algorithm, a novel learning algorithm with alternating self-supervised prediction, which addresses challenges associated with performance across tasks and learning without any specific reward function. Our approach introduces an additional phase, called the *task-agnostic* phase, into the algorithmic structure of Schwarz et al.[13]. This phase complements the existing progress and compress phases. In the progress phase, specific tasks are learned and in the compress phase, the newly learned knowledge is compressed for later reuse. The task-agnostic phase is therefore crucial for building exploration policies and gaining generalization capabilities, which can be reused later in subsequent phases. Within the task-agnostic phase the agent explores its environments in a self-supervised manner, without external goals and then compresses its acquired task-agnostic knowledge. This task-agnostic compressed knowledge is then later used for further task-agnostic exploration, in a periodic manner, aiming at maximizing the pursuit of novel states without extrinsic rewards from the environment. Later when tasks become specific, the agent can leverage this task-agnostic consolidated knowledge for specific tasks. The concept is intuitively perceived as a reflection of the enjoyment derived from the actions, periodically, thereby consolidating the pleasure experienced. The agent explores and adapts to new environments and solves tasks faster, which enables faster knowledge transfer and thus increases the sample efficiency of the agent. We evaluate our algorithm against the baseline approach of Schwarz et al.[13] on Atari 2600 games from the arcade learning environment[16]. To the best of our knowledge, this is the first work on task-agnostic policy distillation in continual reinforcement learning.

The primary contributions of our work are summarized as follows.

1. We develop a novel task-agnostic policy distillation algorithm designed to learn exploratory behaviors without relying on task-specific rewards. This algorithm allows for the transfer of learned exploratory behaviors to a target policy, resulting in faster learning and improved performance on downstream tasks.
2. We develop a novel continual reinforcement learning framework that incorporates a task-agnostic phase along with progress and compress phases. This framework facilitates the learning of novel tasks over time while overcoming catastrophic forgetting in a scalable manner.
3. We evaluate our approach against three different continual learning methods across five reinforcement learning tasks from the Arcade Learning Environment. These experiments were performed in a continual learning setup where tasks are encountered sequentially.
4. In the interest of promoting transparency and reproducibility, we make our code available at https://github.com/wabbajack1/TAPD.

Related work
Policy distillation
Policy distillation, introduced by Rusu et al.[17], serves as a fundamental technique for transferring knowledge from multiple task-specific expert policies into a single, generalized student policy. This approach reduces the computational burden of multi-task learning by compressing multiple models into one, which can perform well across various tasks. Following this, several studies have further explored and refined policy distillation techniques. For instance, Czarnecki et al.[18] examined the broader landscape of policy distillation methods, comparing various approaches and their theoretical underpinnings. They highlighted different formulations, such as entropy-regularized distillation, which allows for faster learning and better convergence properties in diverse situations. Another notable approach is the work by Watkins et al.[19], where policy distillation was employed to incorporate external advice into the learning process. This method allowed for the integration of expert knowledge, enabling the agent to quickly adapt to new tasks and improve overall performance without extensive retraining. Sun et al.[20] introduced real-time policy distillation in deep reinforcement learning, which aimed to distill policies continuously during training. This approach enhanced the adaptability of the learning agent by continuously integrating the distilled knowledge, thereby improving its performance across a range of tasks. These studies underscore the versatility and effectiveness of policy distillation in multi-task reinforcement learning, demonstrating its potential to significantly improve training efficiency and policy robustness across various tasks and environments.

Continual learning
Continual/lifelong learning is the ability to continually acquire, fine-tune, and transfer new knowledge and skills over time. Continual learning agents face the problem of catastrophic forgetting when learning from changing input distributions, which causes the agent to forget old knowledge when learning new tasks. They are also expected to reuse previous knowledge to learn new tasks faster without retraining from scratch or re-accessing previously seen data. This is often referred to as the stability-plasticity dilemma, where stability is the ability to retain old knowledge and plasticity is the ability to acquire new knowledge. Continual learning models can be broadly categorized into three groups: (1) models that regulate intrinsic levels of plasticity[13,21], (2) models that dynamically change their architecture to suit the learning of each individual task[12,22,23], and (3) models that use experience replay for long-term memory consolidation[24,25] (see[26] for a review).

A key limitation of currently established approaches to continual learning is their reliance on static annotated data (e.g., images or texts). However, in more natural learning settings, continual learning models are required to learn continually from sequential data with meaningful temporal relations and with sparse teaching signals (annotations). Recently, a number of approaches have been proposed to achieve unsupervised continual learning, but they have primarily been designed and applied to incremental image classification[27,28]. Consequently, there is a need for novel models that support unsupervised continual learning for decision-making and reinforcement learning problems where task labels are sparse or unavailable. We propose to address this challenge via task-agnostic policy distillation with self-supervised prediction for efficient, continual reinforcement learning.

Mitigating catastrophic forgetting

Kirkpatrick et al.[11] focused on the problem of catastrophic forgetting, where a neural network loses its ability to perform previously learned tasks when trained on new tasks. The authors introduce an algorithm called Elastic Weight Consolidation (EWC) that selectively slows down learning on weights that are important for previously learned tasks. The algorithm is inspired by synaptic consolidation observed in biological brains[29]. Specifically, the algorithm uses a quadratic penalty on the difference between the parameters for the old and new tasks. This penalty slows down the learning process for task-relevant weights that encode previously learned knowledge. This approach helps to preserve previously acquired knowledge. By using the Fisher Information Matrix, EWC effectively balances the need to learn new tasks while retaining performance on old tasks.

Rusu et al.[12] introduced another concept for mitigating catastrophic forgetting with the concepts of Progressive Networks. These networks are designed to retain a pool of pretrained models during training and learn lateral connections from them to extract useful features for new tasks. This architecture is resistant to catastrophic forgetting and enables knowledge transfer across tasks. When a new task is introduced a separate neural network, named *column*, is generated and appended into the Progressive Network. This new column establishes lateral connections with the existing columns. Each column corresponds to a previous task where useful features could be learned from these lateral connections when the new task is introduced. To circumvent the problem of catastrophic forgetting, the parameters associated with the previous tasks are frozen during learning. Therefore, a separate set of parameters is learned specifically for the new task.

To address the problem space in continual RL and unify different methods, Schwarz et al.[13] proposed an approach called "Progress & Compress". This approach combines multiple methods and leverages their complementary strengths within an algorithmic framework. The proposed framework consists of two neural networks: an active network and knowledge base network, which are trained in two distinct alternating phases between consecutive tasks, namely the progress and compress phases, respectively. During the progress phase, the active network utilizes lateral connections (Rusu et al.[12]) from the knowledge base network. The knowledge base network contains distilled knowledge[30] from newly learned tasks while retaining knowledge of old tasks in the parameter space (Kirkpatrick et al.[11]). This regularization ensures that the learning parameters are similar to those adapted to older tasks in the parameter space, resulting in an average performance across all encountered tasks[23]. This approach leverages information from previous tasks to facilitate positive forward transfer. The retention of old tasks is achieved through an online variant of the EWC algorithm, where Schwarz et al.[13] introduced the concept of gracefully forgetting old tasks to free up capacity for new tasks. It also addresses the lack of scalability the EWC has when dealing with a large number of tasks as it requires keeping a separate regularization term for every previous task. This is done by using a running sum of the Fisher information matrices representing the relative importance of weights to older tasks as the only regularizer. The details on the implementation of the "Progress & Compress" algorithm are given in Appendix A in the supplementary file.

The approach proposed by Rusu et al.[12] shows promising results in preventing forgetting of previous tasks in RL. However, a limitation is the increasing number of parameters as the number of tasks grows, leading to increased computational complexity and memory requirements. Additionally, the methods in both Rusu et al.[12] and Kirkpatrick et al.[11] require knowledge of the task label for inference, which may not always be available. This makes it difficult to adapt the architecture to task changes during inference, limiting its applicability in dynamic scenarios. On the other hand, Schwarz et al.[13] addressed several challenges in continual learning, including the ever-growing parameter space by design of their algorithm. They introduced the online variant of EWC to gracefully forget previous tasks and used lateral connections. Nevertheless, none of these approaches has addressed learning without clear task boundaries[14].

Our proposed approach builds on the Progress and Compress framework[13] but extends it to enable continual learning in the absence of clear task boundaries. Additionally, we incorporate lateral connections as in Rusu et al.[12] to reuse features learned from previous tasks. Unlike Rusu et al.[12], which requires storing a separate policy network for each task, our method maintains a single policy network, the knowledge base, and distills knowledge from new tasks into this network, facilitating scalable learning.

Curiosity and intrinsic motivation

As stated by Chentanez et al.[31], motivation is a critical element that comes in two forms: intrinsic and extrinsic. Extrinsic motivation is fueled by specific rewards, whereas intrinsic motivation arises from the inherent enjoyment or curiosity of the activity. Dopamine is a crucial neurotransmitter that influences both motivation and the pursuit of rewards. As suggested by Speranza et al.[1], dopamine operates by binding to receptors in specific brain regions, such as the nucleus accumbens and the prefrontal cortex, thereby affecting motivation and pleasure. Dopamine is not only vital for extrinsically motivated behaviors aimed at obtaining specific rewards but also for intrinsically motivated behaviors focused on exploration and novelty[31]. This mechanism encourages behaviors that are essential for survival from an evolutionary standpoint and also increases the efficiency of learning to solve new problems, i.e. knowledge generalization. Among animals, for instance, novel sensory

stimuli can trigger dopamine cells in a manner similar to unexpected rewards[32]. This activation tends to vanish as the stimuli become familiar, explaining why novelty is rewarding in itself.

Computational frameworks for intrinsic motivation are inspired by how human infants and children set their objectives and gradually develop skills, as highlighted by Parisi et al.[4]. Infants are skilled at playing and have an impressive ability to create new structured behaviors in unstructured environments that do not provide clear extrinsic reward signals. Current advancements in reinforcement learning incorporate elements of curiosity and intrinsic motivation, especially in situations with limited or sparse rewards. In environments with limited extrinsic rewards, an agent can depend on curiosity-driven exploration. As a result, the agent can explore policies and discover novel states more efficiently. Hafez et al.[33] delved deeper into this concept, emphasizing the role of intrinsic motivation in reinforcement learning. They introduced the Intrinsically-motivated Continuous Actor-Critic (ICAC) algorithm, a curiosity-driven RL approach that incrementally builds a network of local forward models. These models assist in computing the agent's intrinsic rewards. Additionally, the algorithm employs an Instantaneous Topological Map (ITM) to partition the sensory space, guiding the agent towards information-rich states and actions. The research indicates a significant performance gain for the intrinsically motivated agent compared to agents that are only motivated by extrinsic rewards.

Pathak et al.[7] proposed an approach to curiosity-driven exploration where curiosity is formulated as the error in an agent's ability to predict the consequences of its actions in a visual feature space, in a self-supervised manner. In this way, exploration is promoted by visiting states that are difficult to predict, similar to Schmidhuber[34]. The visual feature space is learned through a self-supervised inverse dynamics model. The agent is structured with two primary subsystems: a reward generator and a policy. The reward generator produces an intrinsic reward signal based on the prediction error of the agent's knowledge about its environment. The Intrinsic Curiosity Module (ICM), as the reward generator, consists of two neural network models: the inverse and the forward models. The inverse model aims to predict the agent's action based on the feature encodings of two consecutive states, while the forward model predicts the feature encoding of the next state based on the current state's feature encoding and the taken action.

Building upon the ICM approach, the Intrinsic Sound Curiosity Module (ISCM) developed by Zhao et al.[15] utilizes the power of sound in robotic actions. The ISCM provides feedback to the agent based on crossmodal prediction error, allowing them to develop robust representations and efficient exploration. This approach has shown scalability to high-dimensional input and leverages prior knowledge to accelerate learning of downstream tasks. Zhao et al.[15] emphasized the ability of the approach by Pathak et al.[7] to scale to high-dimensional input and utilize knowledge from past experiences for more efficient exploration and learning of unseen tasks, therefore making ICM as a reliable tool for exploring environments.

Background
Reinforcement learning

Let us consider a standard RL problem in which an agent interacts with a complete observable environment and adopts a strategy to maximize the cumulative future reward. An environment consists of a state space S, an action Space A, a reward function $r : S \times A \rightarrow R$, a dynamics model $p(s_{t+1}|s_t, a_t)$, and a discount factor $\gamma \in [0, 1]$. Therefore, an RL problem is precisely described as a Markov Decision Process (MDP). Let $\pi : S \rightarrow P(A)$ be the policy, a mapping from states to probability distributions over actions. At each timestep t, the agent takes an action $a_t \sim \pi(s_t)$ and receives a reward $r_t = r(s_t, a_t)$ while the environment transitions into a new state $s_{t+1} \sim p(\cdot|s_t, a_t)$. A discounted sum of future rewards defines the return $R_t = \sum_{i=t}^{T-1} \gamma^{i-t} r(s_i, a_i)$. The goal is to maximize the expected return $J = \mathbb{E}_{s_0 \sim S_0}[R_0|s_0]$, where $S_0 \subseteq S$ is a set of initial states.

The action-value function is defined as $Q_\pi(s_t, a_t) = \mathbb{E}[R_t|s_t, a_t]$, and the optimal policy π^* satisfies $Q_{\pi^*}(s, a) \geq Q_\pi(s, a), \forall(s, a) \in S \times A$. When the model is not available, the optimal Q-function is approximated by a neural network with parameters θ^Q and trained to minimize the loss $\mathcal{L}$ between the target value $y_t = r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a|\theta^{Q'})$ and the current Q-estimate, where $\theta^{Q'}$ are the target Q parameters and are updated slowly towards θ^Q [35,36]:

$$\mathcal{L}(\theta) = (y_t - Q(s_t, a_t|\theta^Q))^2 \tag{1}$$

In RL, we can directly optimize a policy π that is parameterized by θ, with the aim of maximizing the expected return i.e. updating θ in the direction of an estimate of the gradient $\nabla \log \pi(a_t|s_t; \theta) R_t$. Of particular interest is the actor-critic methods, which learn a policy and a value function simultaneously, such as Advantage Actor Critic (A2C)[37,38].

Advantage actor critic (A2C)

In the landscape of Reinforcement Learning (RL), the Advantage Actor Critic (A2C) algorithm occupies a prominent role, offering an elegant and efficient way to balance between action evaluation and policy optimization[38]. A2C is a synchronous variant of the A3C algorithm, originally introduced in Mnih et al.[37]. It falls under the category of policy gradient methods and employs an on-policy value function (the critic), denoted as $V_\theta^\pi(s_t)$, as its training baseline. This critic function, although adding a bias, substantially reduces the variance of the policy gradient estimates, leading to quicker and more stable training.

The backbone of A2C lies in the integration of the advantage function and the synchronous training across multiple, disjoint agents, each operating in distinct environments to accumulate training samples. The intuition behind the use of the advantage function in policy gradient methods is that a step in

the policy gradient direction increases the probability of actions that are better than average, while suppressing the likelihood of suboptimal actions[39]. This behavior is encapsulated by the advantage function $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V_\theta^\pi(s_t) = r_t + \gamma V_\theta^\pi(s_{t+1}) - V_\theta^\pi(s_t)$, which measures how the selected action a_t compares to the default behavior of the policy in state s_t. Consequently, when updating the policy, the term $\nabla_\theta \log \pi_\theta(a_t|s_t) A_t$ will point in the direction that enhances $\pi_\theta(a_t|s_t)$. The stochastic policy π (also called the actor) is updated using stochastic gradient ascent. The gradient estimator for the policy is therefore

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^\pi(s_t, a_t) \right] \tag{2}$$

Subsequently, the policy parameters θ are updated as $\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_\theta)$, where α is the learning rate and k is the iteration number. The Advantage Actor-Critic (A2C) algorithm uses two neural networks with shared parameters, an Actor and a Critic, to perform decision-making tasks. In each episode, the agents choose actions in parallel based on the Actor's policy, execute them to obtain rewards, and then compute an advantage metric to measure how well those actions did compared to the average. Both the Actor and Critic are then updated based on this information. Since A2C samples actions according to the current policy and then updates the policy based on those sampled actions, it is considered an on-policy algorithm.

Continual learning

Lifelong learning or Continual Learning refers to the ability of a system to continually acquire, fine-tune, and transfer knowledge over an extended period of time. This capability is essential for computational systems and autonomous agents that interact with changing environments, i.e. dynamic data distributions[40]. However, the challenge lies in avoiding catastrophic forgetting, where the acquisition of new knowledge interferes with previously learned information, also rephrased as the stability-plasticity dilemma. This dilemma concerns the balance a system must maintain between its ability to learn new information (plasticity) and its need to retain existing knowledge (stability). Two types of plasticity are essential for a stable, continuous lifelong learning process: Hebbian plasticity for positive feedback and compensatory homeostatic plasticity for neural stability[4,41].

In formal, continual learning, as described by Zeno et al.[14], a continuously learning algorithm is confronted with a task sequence without the possibility of accessing data from past or future tasks. Specifically let $\mathcal{L} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$ be the continual learning space, where $\mathcal{L}$ represents the continual learning process, $\mathcal{T}_i$ represents the ith task to be learned and n is the total number of tasks. The objective is to optimize the loss function J across all tasks while ensuring that the model does not forget the older tasks when learning new ones: $\min_\theta \sum_{i=1}^{n} J(\theta; \mathcal{T}_i)$. Subject to the constraint that the model parameters θ are shared across all tasks and updated incrementally $\theta_{k+1} = \theta_k - \alpha \nabla J(\theta_k; \mathcal{T}_i)$, where k is the iteration step of the parameters θ.

In continual learning, it is important to differentiate in which context the agent should solve problems, because there could be different variations in which the agent encounters problems and in which granularity the continual learning agent is solving a task. Van de Ven and Tolias[42] suggested distinct scenarios for continual learning to standardize evaluation and enable more meaningful comparisons across different methods. In *Task-Incremental Learning*, models are always informed about the task at hand, allowing for task-specific components in the model. Here the model is explicitly instructed which task to perform each time, enabling it to use specialized settings or tools optimized for each task. Whereas in *Domain-Incremental Learning*, task identity is not available at test time. However, models only need to solve the current task, they are not required to infer which task it is. This scenario is relevant for situations where the structure of tasks remains the same, but the input distribution changes. In *Class-Incremental Learning*, models must solve each task seen so far and infer which task they are presented with. *Task-agnostic learning* expands on the scenario of Domain-Incremental Learning by eliminating the use of task IDs during the training phase (Zeno et al.[14]). A task-agnostic agent is capable of learning in an environment without any extrinsic goal[8], meaning no specific reward function is given during training. Specifically, this means that the agent can solve different tasks without knowing their boundaries or task-IDs, i.e. indicating which task they are solving.

Task-agnostic policy distillation

In knowledge distillation[17,30], the goal is to train a target network, referred to here as the knowledge base, to produce the same output distribution as the original network, referred to here as the active column. In the proposed task-agnostic phase, the active column is initially trained to maximize an intrinsic reward that encourages exploration, without relying on task-specific extrinsic rewards. Subsequently, the knowledge from the active column is transferred to the knowledge base by distilling the exploratory behavior of the trained active column network into the knowledge base network. This process is repeated iteratively as necessary. Below, we provide more details on the training environment and the variations of the task-agnostic phase, followed by a discussion on the intrinsic reward used for training.

Meta-environment

Let $\mathcal{E}$ represent the set of all environments. For a given task k, let $E^k \in \mathcal{E}$ denote the environment E with task k, where $k \in \mathbb{N}$. Within the context of continual learning, specifically in the Atari 2600 domain[16], it is often assumed that each environment is considered a distinct task. This concept extends to the idea of learning task k within a given environment E and $r_{t+1}, s_{t+1} \sim P^{E^k \in \mathcal{E}}(\cdot)$ represents the dynamics of that environment. This

formalism is congruent and applicable to scenarios where different tasks exist within a single environment, such as a robot solving different tasks in an environment, where E^k signifies a task within E. In the case of the Atari 2600 domain[16], we consider all games/environments to constitute one unified environment and conceptualize $\mathcal{E}$ as a Meta-Environment, dependent upon the task context and the definition of what constitutes a task, given the versatile nature of the term task. The Meta-Environment replicates a condition where a single environment is present, encompassing diverse tasks. This concept is consistently applied throughout the paper, presenting inherent robustness to changes in environmental dynamics, such as the sudden introduction of a new game, which are interpreted as shifts within the Meta-Environment and therefore do not negatively impact the task-agnostic policy (see section "Variations of the task-agnostic phase"). For a visual representation, see Fig. 1. It is important to note that the tasks can be sampled in a predefined, specified sequence from the Meta-Environment. However, this specified sequence can be relaxed to represent a specific distribution, from which the tasks (game environments) are sampled. Learning within the Meta-Environment can be more challenging due to the different textures of the tasks (i.e. game environments) and the high dissimilarity between tasks.
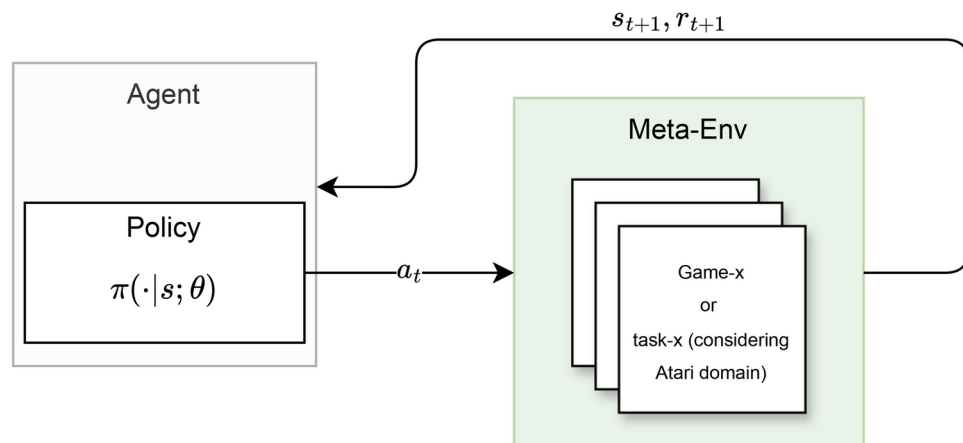
Variations of the task-agnostic phase

As proposed by Schwarz et al.[13], the generality of the introduced progress and compress algorithm makes it a suitable candidate for integration into various frameworks. The work of Schwarz et al.[13] is extended to include a *task-agnostic* phase. Consequently, this paper introduces the *Task-Agnostic Policy Distillation* framework, which augments the progress and compress framework. Building on the foundational architecture of Schwarz et al.[13], the task-agnostic phase introduces variations that enable the agent to operate without explicit task boundaries[14]. This phase implies a learning period where the agent learns without an external goal, meaning it receives no rewards from the environment, addressing the question of learning without a well-structured reward function. A curiosity-driven intrinsic reward signal is introduced through self-supervised prediction (see section "Intrinsic reward"), as conducted by Pathak et al. and Zhao et al.[7,15]. The task-agnostic phase develops its exploratory policy by maximizing the RL objective $\mathbb{E}_\pi\left[\sum_{n=0}^{\infty} \gamma^n r_{t+n}^i\right]$, where r_t^i represents an intrinsic reward generated by the agent itself at timestep t. This increases the probability of discovering novel states and potentially acquiring novel skills. The agent can leverage its exploratory policy and the generalized skills it induces to become a versatile problem solver when faced with specific tasks, addressing issues of positive forward transfer and enhancing sample efficiency. Specifically, the loss to be minimized in the task-agnostic phase is therefore:

$$\mathcal{L}^{agnostic}(\theta) = -\mathbb{E}_\pi\left[\sum_{n=0}^{\infty} \gamma^n r_{t+n}^i\right]. \tag{3}$$

Variation 1 (TAPD)

Variation 1 of the task-agnostic phase is the most versatile among all the variations discussed in this work. It is suitable for diverse environments in which the action space remains consistent but the tasks vary, where the environments are conceptualized as tasks by the Meta-Environment, a collection of different game environments. In this variation, tasks (game environments) are sampled based on a specific distribution from the Meta-Environment, such as uniform sampling here. This selection mechanism implies that the agent lacks awareness of when task transitions occur, reflecting the task-agnostic scenario outlined by Zeno et al.[14]. As depicted in Fig. 2, this variation builds upon the framework established by Schwarz et al.[13]. The architectural designs within the progress and compress phases align with the baseline[13]. The task-agnostic phase also makes use of the active column. The first variation of the task-agnostic phase can be described as a process that leverages the A2C algorithm to minimize $\mathcal{L}^{agnostic}(\theta)$. In this phase, the exploratory policy, i.e. the task-agnostic policy, is distilled into the knowledge base (KB) after a specified number of intermediate timesteps, followed by the

Fig. 1. Illustration of the meta-environment, representing different game environments as tasks within the context of Atari 2600 games.
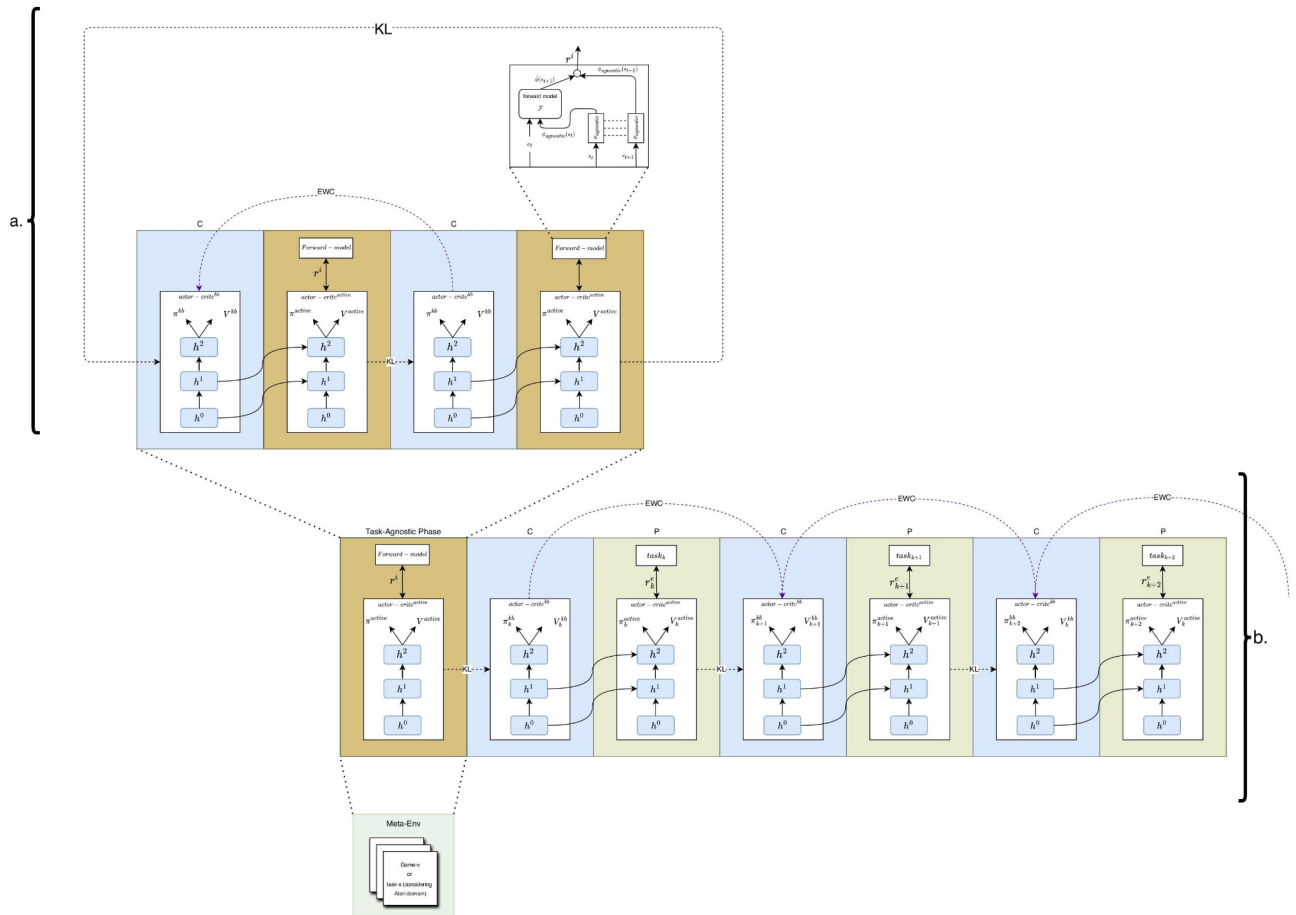
Fig. 2. Overview of our task-agnostic policy distillation framework. (**a**) The task-agnostic phase is an abstraction of a process where intermediate alternations between maximizing intrinsic rewards and distillation occur. This process follows the same alternating pattern as in the progress and compress framework. (**b**) Here, the task-agnostic phase is initially used before alternating between progress (P) and compress (C) phases. When considering the Atari domain, each task can be randomly selected from the Meta-Environment, therefore simulating one game environment. In the C phase, the recently learned policy by the active column (green) is distilled into the knowledge base (KB) (blue) using the KL loss between the active column and KB while protecting KB's old values using Elastic Weight Consolidation (EWC). In the P phase, features learned from previous tasks are reused via lateral connections when learning new tasks. r_k^e and r^i are the extrinsic reward of task k and the task-independent intrinsic reward, respectively. h is a hidden layer.

selection of another game environment, highlighting the agent's unawareness of task transitions. Thus, this variation of the task-agnostic phase encompasses learning through both curiosity and distillation (see part (a) in Fig. 2). The distilled task-agnostic policy is then used through the lateral connections to learn a new task-agnostic policy. This new policy is once again distilled and used to enhance the knowledge base with elements of curiosity. The act of distilling the exploratory distribution of one policy into another augments the exploratory behavior, as distillation has more information per training sample than hard targets[30]. The integration of various task-agnostic policies into the knowledge base is performed in an alternating manner, i.e.

$$\xrightarrow[x-steps]{\text{explore/train}} \min \mathcal{L}^{agnostic}(\theta) \xrightarrow[x-steps]{\text{compress}} \theta^{kb} \xrightarrow[x-steps]{\text{explore/train}} \min \mathcal{L}^{agnostic}(\theta) \xrightarrow[x-steps]{\text{compress}} \ldots \quad (4)$$

where θ and θ^{kb} are the learning parameters of the active column and knowledge base, respectively.

This process is illustrated by the loop in Fig. 2, within the abstraction of the task-agnostic phase (see part (a) in Fig. 2). It is noteworthy that the Online EWC algorithm is employed throughout the process, ensuring the retention of knowledge from previous exploratory policies. Since the task-agnostic phase solely relies on the intrinsic reward r_t^i for learning, the agent does not require specific reward schemas, eliminating the need for any task-ID during the training process. This phase is suitable for sparse-reward environments as it does not require extrinsic rewards, making it an ideal pre-training step for downstream tasks. Once exploration is completed for a given number of steps, the Progress and Compress algorithm proposed by Schwarz et al.[13] is used for tasks that are

designed to maximize extrinsic rewards (see part (b) in Fig. 2). The progress phase uses generalized knowledge from task-independent strategies to accomplish specific tasks. An analogy for this task-agnostic phase can be made with a child exploring different rooms. After exploring a room, the child consolidates the new experiences and knowledge gained (distillation). By exploring various rooms and consolidating their understanding each time they enter a new one, the child is subsequently assigned specific tasks by their parents.

Variation 2

The second variation relies on configuring the general Variation 1 of the task-agnostic phase. This means configuring the Meta-Environment in such a way that it only has one task (i.e., Atari game environment) to explore. Therefore, within the task-agnostic phase, only one game environment is being distilled in alternating phases. After each intermediate timestep, the distillation process begins again, followed by exploration, as described in Sequence 4, but without selecting different game environments after each intermediate step. This reflects the case where we only have one environment with different tasks, unlike the Atari domain where one environment has only one task, such as scoring as many points as possible.

Variation 3

The third variation relies on configuring Variation 2 of the task-agnostic phase. This involves setting the number of timesteps to only run for a specific number of steps and compressing the task-agnostic policy only once, before adhering to the standard protocol of the Progress and Compress method, proposed by Schwarz et al.[13]. This follows the steps (the downstream task loss $\mathcal{L}^{progress}(\theta)$ is given in Appendix A in the supplementary file):

$$\xrightarrow[x-steps]{\text{explore/train}} \min \mathcal{L}^{agnostic}(\theta) \xrightarrow[x-steps]{\text{compress}} \theta^{kb} \xrightarrow[y-steps]{\text{train}} \min \mathcal{L}^{progress}(\theta) \xrightarrow[y-steps]{\text{compress}} \dots \tag{5}$$

This approach relaxes the requirement for alternating distillation of task-agnostic policies, as only one task-agnostic policy would be distilled into the knowledge base. An analogy can be drawn with an agent that explores one room, acquiring skills within it that can later be applied to different rooms.

Intrinsic reward

The task-agnostic agent predominantly utilizes self-supervised prediction as its learning mechanism. By learning to predict future states from current states and actions, the agent is able to navigate and comprehend its environment, eliminating the need for extrinsic rewards. This intrinsic motivation, inspired by the ICM module of Pathak et al.[7], is driven by the prediction error of a forward dynamics model and enables the agent to explore and acquire knowledge about its surroundings, preparing it for future tasks in diverse environments. A distinction from Pathak et al.[7] is made by excluding the inverse dynamics model of the ICM module, as this work is solely focused on learning a forward model. In alignment with Pathak et al.[7], it is advantageous to make predictions about the next state within the feature space. Hence, let $\phi_{agnostic}$ be the visual encoder (here, a convolutional neural network) and let $\mathcal{F}$ be a multi-layer perceptron. If the state at timestep t is represented as s_t (raw pixels) and the action as a_t, then $\mathcal{F}(\phi_{agnostic}(s_t), a_t) = \hat{\phi}(s_{t+1})$, where $\hat{\phi}(s_{t+1})$ is the predicted feature vector of the next state. Therefore, the composition of the networks $\mathcal{F}$ and $\phi_{agnostic}$ forms the ICM module, excluding the inverse model. The loss used to optimize the forward dynamic model is defined as an L_2 norm of the difference between the observed and predicted feature vectors of s_{t+1} i.e. $\mathcal{L}^{forward} = \left\| \phi_{agnostic}(s_{t+1}) - \hat{\phi}(s_{t+1}) \right\|_2^2$.

The overall intrinsic reward used to seek novel states is $r_t^i = \log(\mathcal{L}^{forward} + \epsilon)$, where ϵ is a constant added to maintain numerical stability, for values near zero.

Using the error in predicting future states as an intrinsic reward results in exploratory behavior that enables the agent to explore and acquire knowledge about its surroundings without the need for external supervisory signals, relying only on its curiosity. By encouraging visits to states that are difficult to predict (high prediction error areas), this approach fosters a directed exploration strategy aimed at improving the agent's forward model. This strategy is particularly effective in sparse-reward environments, as it does not depend on extrinsic rewards. During the task-agnostic phase, we apply this principle to learn an exploratory policy that is periodically distilled into a knowledge base. After each distillation, a new game environment is selected where the agent further refines its task-agnostic policy using intrinsic rewards and lateral connections from the knowledge base, which now includes the recently distilled policy. This new policy is once again distilled and used to enhance the knowledge base. The distillation of the exploratory action distribution of one policy into another augments and diversifies the exploratory behavior. After the task-agnostic phase, learning downstream tasks in the subsequent progress and compress phases is accelerated due to the generalized knowledge from task-agnostic exploration stored in the knowledge base.

Experimental evaluation

In this section, we present the experimental evaluation of our approach. First, section "Environments" describes the Atari games that constitute the Meta-Environment to which our approach and the compared methods are applied. Next, section "The intrinsic reward in the task-agnostic phase" details the implementation of the task-agnostic phase and demonstrates the agent's performance and learning progress during this phase. Then, section "Evaluation of knowledge accumulation and transfer to downstream tasks" compares the performance of all methods on downstream tasks in terms of learning efficiency assessed using the average game score and the entropy of the policy distribution during the progress phase, where tasks are encountered sequentially. Following

this, section "Assessing forward transfer" evaluates forward transfer as an indicator of adaptability by analyzing the average score on each subsequent visit to the considered tasks. Finally, section "Computational efficiency" evaluates the computational efficiency of each method, highlighting the trade-offs between performance and resource demands.

Environments

As in Schwarz et al.[13], the task-agnostic policy distillation framework is used in the Atari domain. In this paper, we focuses on five different Atari games: Pong, SpaceInvaders, BeamRider, DemonAttack, and AirRaid. The Meta-Environment combines these five game environments. To modify the action space, a custom action wrapper is employed. This wrapper maps the intended actions of the agent to the corresponding movements within the specific game environment. The action space is downscaled, removing unnecessary actions, as suggested by Kanervisto et al.[43]. Table 1 illustrates how actions are mapped to different games/tasks. This mapping can vary depending on the specific requirements of each game/task. For instance, in the case of BeamRider, the agent selects action 2 from the policy distribution. However, the game interface executes action 3, as specified in BeamRider. As a result, the actual (game-specific) action space is hidden from the agent.

These modifications to the action space prevent the attempt of "pointless" actions. By reducing the size of the action space, the complexity of the problem is decreased, requiring fewer computational resources and potentially speeding up training time.

Reward scaling

Due to the varying value ranges of rewards in different game environments/tasks, enhancing stability across a series of sequentially learned tasks is crucial in all environments. To address this, a single modification is made to the reward structure of the games, but only during the training phase. Given the substantial variability in the scale of scores from game to game, all positive rewards are set to be 1, and all negative rewards to -1, with 0 rewards remaining unchanged. This normalization facilitates more stable and consistent learning across varying tasks and game environments by mitigating the impact of extreme reward values, as demonstrated in Mnih et al.[35].

Observations

At each timestep, an agent only receives normalized visual input, which is represented as a 84×84 grayscaled image. The environment is configured for frame skipping, allowing the agent to interact less frequently with the environment. Observations from four consecutive frames are stacked together to form an observation with enhanced temporal dimensionality. Consequently, the agent's policy receives input in the format $[N, 4, 84, 84]$, where N represents the number of environments being used in parallel for training the agent in the progress phase with the A2C algorithm. Our implementation of A2C is based on PyTorch[44,45] and incorporates concepts from Raffin et al.[46] and Lucchesi et al.[47]. We used Bayesian Hyperparameter Optimization over selected configurations to maximize normalized scores across tasks. Our experiments indicate that TAPD demonstrates strong stability, with minimal performance variation across different hyperparameter settings and consistent behavior across configurations. Details on the hyperparameters of the learning architecture, algorithms, and experimental settings are provided in Appendix B in the supplementary file.

The intrinsic reward in the task-agnostic phase

We evaluate the performance of the agents based on task rewards, which are specifically designed to measure active interactions. Although extrinsic rewards are recorded, they are not used during the task-agnostic training phase. Instead, the agent focuses exclusively on optimizing the intrinsic rewards it generates. The task-agnostic phase receives tasks from the Meta-Environment, which includes BeamRider and SpaceInvaders. These tasks are uniformly sampled from the Meta Environment without any task boundaries[14]. We employ TAPD's task-agnostic phase, as it is the most general of all the three variations.

Figure 3 illustrates the learning progress of TAPD during the task-agnostic phase. In this phase, the task-agnostic policy is compressed after a specific number of timesteps into the knowledge base. As shown in the figure, the agent's performance improves even without extrinsic rewards. Every 300k timesteps, the task-agnostic policy is distilled into the knowledge base, resulting in an exploratory repertoire. This exploratory knowledge is then utilized within the task-agnostic phase, using generalized knowledge to further generalize knowledge. When a task change occurs, the active column in the task-agnostic phase quickly adapts to the new environment, as indicated by the spikes in the graph. This suggests that the knowledge base is aware of the environment it is in, having learned the dynamics of the environment through the forward model that produces the intrinsic rewards that guide the action selection of the active column. These rewards aid in further exploration of the environment, leading to the discovery of more novel states. The rapid task adaptation also highlights the mechanism of the knowledge base, as the active column remembers the previous situation without external goals from the

Game/Task	NOOP (0)	FIRE (1)	RIGHT (2)	LEFT (3)
Pong	0	1	2	3
BeamRider	0	1	3	4
SpaceInvaders, DemonAttack, AirRaid	0	1	2	3

**Table 1.** Action mapping for different games/tasks.
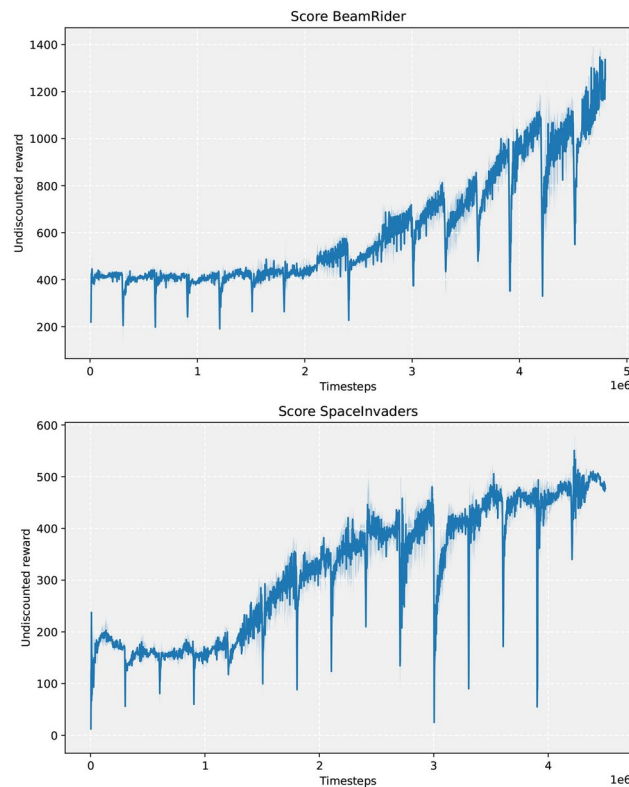
Fig. 3. Performance evaluation in the task-agnostic phase. The environment is uniformly sampled, indicating no task-boundaries. Runs averaged over 8 random seeds. Timesteps=300,000 between distillation rounds in the task-agnostic phases. Averages are taken over 100 episodes.

environment with the help of the knowledge base. After the maximum number of timesteps in the task-agnostic phase has been reached, the alternation between progress and compress phases begins, but now with a pre-explored knowledge base, which accelerates the learning process for downstream tasks.

Evaluation of knowledge accumulation and transfer to downstream tasks

We compare our Task-Agnostic Policy Distillation (TAPD) algorithm with its task-agnostic phase, Online EWC[11,13], Progressive Nets[48], and the Progress & Compress baseline from Schwarz et al.[13]. This comparison aims to evaluate whether the task-agnostic phase can accelerate transfer learning in the progress phase for downstream tasks, while also improving sample efficiency. The task-agnostic phase, as depicted in Fig. 2, serves as the pre-training phase. The algorithms share the same architecture and hyperparameters, with the exception that TAPD has specific parameters for the task-agnostic phase. The tasks Pong, SpaceInvaders, BeamRider, DemonAttack, and AirRaid are sequentially learned. The evaluation focuses on the learning performance in the progress phase, as the emphasis lies on evaluating the positive forward transfer and sample efficiency of the active column. In the task-agnostic phase, TAPD uniformly samples SpaceInvaders and BeamRider, simulating tasks with undefined boundaries Zeno et al.[14]. On each visit to a task, the active columns parameters and lateral connections are re-initialized.

Figure 4 shows the learning curves of Pong, SpaceInvaders, BeamRider, DemonAttack, and AirRaid along with their corresponding entropies. In the case of Pong, although TAPD hadn't previously learned the game, its knowledge base enabled much faster exploration of newly encountered tasks, resulting in significantly higher performance than all other algorithms, which showed only slight improvement after 1.5 million timesteps. On the other hand, TAPD already learned to adapt to the tasks at around 0.6 million timesteps, showcasing the high sample efficiency of the proposed approach. This adaptation is also evident in the entropy of the distribution for solving the Pong task. The entropy initially starts lower than the Progress & Compress baseline, indicating that the agent is aware of the required movements. However, the Progress & Compress baseline exhibits a similar pattern throughout the training process, albeit with lower performance compared to TAPD. In later timesteps, TAPD exhibits a more exploratory behavior compared to the Progress & Compress baseline, with occasional decreases. The knowledge base of TAPD seems to retain its exploratory behavior during the learning process of Pong. The entropy in Progressive Nets exhibits a steep learning curve during the second visit, which is expected. This is because the dedicated column used to train the Pong task in the first visit retains and continues to refine the parameters associated with Pong during the second visit, leading to rapid progress, which extends until the third and final visit. The Online EWC algorithm demonstrates a recurring pattern in the initial timesteps of each visit, where the score increases, indicating that the model detects the task but is unable to further improve upon it.
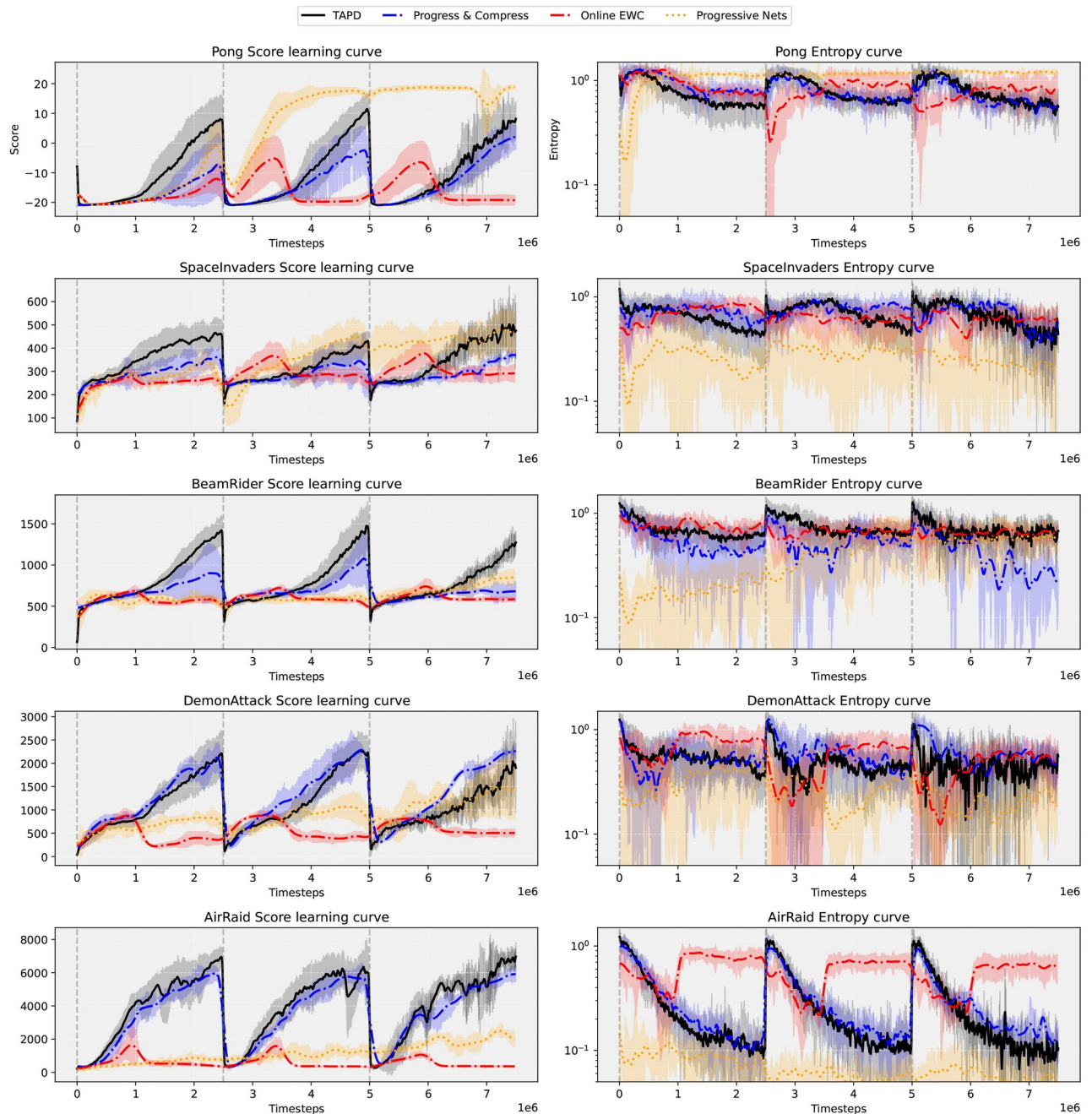
Fig. 4. The learning curves depicted represent the obtained rewards in the progress phase, against Task Agnostic Policy Distillation (TAPD), Online EWC, Progressive Nets, and the reproduced Progress & Compress baseline. Reading from left to right, both performance and entropy are plotted. Tasks are learned in a sequential manner in the following order: Pong, SpaceInvaders, BeamRider, DemonAttack, and AirRaid. TAPD utilizes the distilled knowledge from the task-agnostic phase. Results are averaged over 4 seeds and reflect the averages of scores taken across 100 episodes. Each task is revisited three times (gray vertical lines), allowing for training for 2.5 M timesteps on each visit in the progress phase.

Moving on to the SpaceInvaders task, TAPD appears to remember this environment, which potentially contributes to its superior performance compared to all other algorithms in the initial phase. Throughout the training process, TAPD maintained lower entropy compared to the Progress & Compress baseline, which showed much higher entropy across all three visits. Despite the Progress & Compress baseline employing lateral connections in subsequent visits, its performance does not improve over time due to the high entropy of its policy, which suggests ongoing exploration. This does not mean the baseline is stuck, it may improve its performance after many more timesteps, but this would indicate lower sample efficiency. A similar behavior is observed with the Online EWC algorithm. The Progressive Nets algorithm demonstrates a strong initial performance with no

interference from task-specific learned parameters, indicating a robust growth mechanism. However, despite the lack of interference in Progressive Nets, the TAPD algorithm ultimately converges to a higher performance by the end of the task visits. This suggests a positive forward transfer and a recurring pattern of stable entropy, contributing to its success.

In the BeamRider task, the TAPD algorithm consistently outperforms all other algorithms throughout the visits. Despite the Progressive Nets algorithm's use of dedicated task parameters, its performance and sample efficiency remain low, suggesting that lateral connections between columns and dedicated task parameters do not necessarily enhance forward transfer. In the AirRaid task, the Progress & Compress baseline shows comparable performance to TAPD during the initial visits, but TAPD ultimately surpasses it by the end of the training. This suggests that the distillation of the intrinsically motivated behavior during the task-agnostic phase serves as a strong regularizer, enabling TAPD to achieve higher final performance. The smaller gap between visits in catching up to earlier performance suggests greater sample efficiency. In contrast, Progressive Nets algorithm struggles to catch up with the final performance of TAPD, further highlighting the advantages of the TAPD approach. In terms of entropy, it is evident that the TAPD algorithm begins with exploratory behavior that gradually decreases over the course of visits. This pattern suggests that the agent initially explores the environment more extensively compared to the more conservative action selection process of the Progressive Nets, which may contribute to its slower rate of performance improvement. Injecting task-agnostic behavior into the knowledge base during the task-agnostic phase appears to lead to a better trade-off between exploration and exploitation, enhancing overall performance and demonstrating that task-agnostic policy distillation indeed facilitates positive forward transfer.

Moreover, TAPD is significantly more scalable than Progressive Nets. TAPD requires only two networks, while Progressive Nets face a major limitation: as tasks are added, the network size increases substantially. Specifically, the number of hidden units and feature maps in Progressive Nets grows linearly with the number of columns, and the number of parameters grows quadratically. This scalability issue makes TAPD a more efficient and practical solution, especially when dealing with a large number of tasks. Additionally, Progressive Nets require the specific task ID during training to query the correct column in subsequent visits. This dependency on task identification is not necessary in TAPD, which makes it a more versatile method. TAPD can quickly adapt to tasks during the initial phases of training without needing explicit task IDs, further underscoring its flexibility and robustness compared to Progressive Nets.
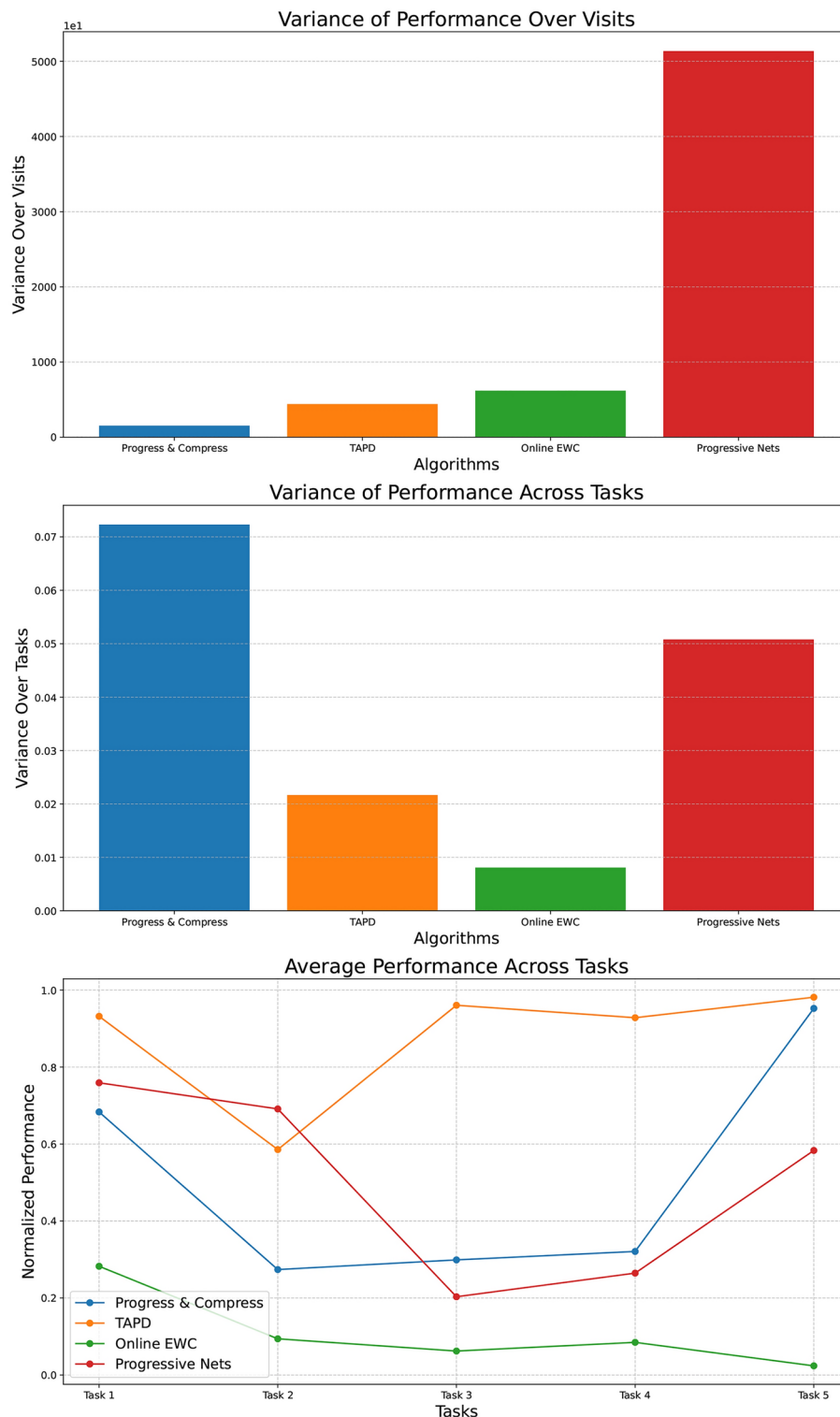
Assessing forward transfer

Positive forward transfer refers to improved performance on a new task immediately following the learning of previous tasks, indicating an algorithm's ability to leverage prior knowledge effectively. Two key indicators demonstrate positive forward transfer: (1) the performance of the active column across tasks over multiple visits, which captures improvement in learning a new task after exposure to prior tasks as well as gains from repeated visits to an old task, and (2) the average normalized performance of each algorithm across tasks, with high values suggesting successful application of previously acquired knowledge. Table 2 presents the performance data of the active column network across different tasks over three visits, comparing TAPD, Online EWC, Progressive Nets, and Progress & Compress. This comparison evaluates how learning on one task influences subsequent task performance. In the table, an upward arrow (↑) indicates improved performance compared to the previous visit, while a downward arrow (↓) indicates decreased performance. Additionally, Fig. 5 illustrates the performance trends across multiple tasks and visits. Performance was evaluated using normalized scores, with further analysis focusing on the variance in performance across tasks and visits. As shown in Table 2, TAPD outperforms all other algorithms in the three visits for the majority of tasks. In the third visit, it maintains the highest scores for

	Task 1	Task 2	Task 3	Task 4	Task 5
First Visit					
Progress & Compress (Active-Col)	− 4.25	327.41	732.12	1029.89	2468.09
TAPD (Active-Col)	14.86	**439.62**	**1441.82**	**2479.65**	**2605.39**
Online EWC	12.14	248.85	528.377	209.62	505.30
Progressive Nets	**16.94**	437.3	646.17	910.16	1279.05
Second Visit					
Progress & Compress (Active-Col)	14.56 ↑	335.97 ↑	877.66 ↑	963.57 ↓	2529.31 ↑
TAPD (Active-Col)	**18.51**↑	**472.91**↑	**1537.52**↑	**2699.95**↑	**2580.15**↓
Online EWC	− 14.71 ↓	275.79 ↑	606.64 ↑	460.08 ↑	433.57 ↓
Progressive Nets	− 3.4 ↓	618.7 ↑	697.6 ↑	612.37 ↓	1455.93 ↑
Third Visit					
Progress & Compress (Active-Col)	15.04 ↑	387.19 ↑	892.40 ↑	1034.01 ↑	2501.98 ↓
TAPD (Active-Col)	19.94 ↑	**483.87**↑	**1550.30**↑	**2383.18**↓	**2506.41**↓
Online EWC	− 17.19 ↓	326.31 ↑	641.08 ↑	592.97 ↑	390.29 ↓
Progressive Nets	**20.33**↑	457.5 ↓	864.68 ↑	1084.24 ↑	2312.06 ↑

Table 2. Performance of the active column network on subsequent tasks after visiting previous tasks. Tasks 1-5 were visited in the following order: Pong, SpaceInvaders, BeamRider, DemonAttack, and AirRaid, respectively. Results are averaged over 8 seeds.

Fig. 5. Analysis of algorithm performance: Assessing forward transfer through variance across visits and tasks and average performance across tasks. Averaged over 8 seeds.

Task 4 and Task 5, despite a slight decline in performance compared to the previous visit, as was also observed for Task 5 in the second visit.

The variance across visits (top plot in Fig. 5) illustrates the variance of each algorithm's performance across multiple visits, providing insight into the stability of their learning processes. A high variance indicates that the algorithm's performance fluctuates significantly from one visit to the next, suggesting instability or sensitivity to specific conditions during each visit. Conversely, low variance indicates a more stable learning process, with

the algorithm performing consistently and similarly across different visits. Progress & Compress demonstrates the lowest variance, reflecting a stable performance across visits, making it potentially more reliable in scenarios requiring consistent outcomes. In contrast, the Progressive Nets algorithm shows the highest variance over visits, indicating significant performance fluctuations and suggesting overfitting during each task visit.

The variance across tasks (middle plot in Fig. 5) measures how consistently each algorithm performs across different tasks. Progress & Compress and Progressive Nets exhibit higher variance, indicating that their performance is uneven across tasks. This suggests that these algorithms may excel in certain types of tasks but struggle with others, leading to a less predictable overall performance. High variance is a sign of overfitting to specific task characteristics, limiting the algorithms generalization capabilities across diverse tasks. This outcome is expected, as the model's complexity increases with each new task and training iteration in Progressive Nets. In contrast, low variance suggests that the algorithm performs more uniformly across different tasks, which is desirable for generalization. TAPD, with its lower variance, demonstrates a more balanced and consistent performance across different tasks, indicating better generalization.

The bottom plot in Fig. 5 provides an overview of the average normalized performance of each algorithm across different tasks, serving as a measure of forward transfer. This comparison highlights how well each algorithm performs relative to the others on the same task. Forward transfer represents an algorithm's capability to leverage knowledge from previous tasks and visits to improve performance on new ones. An upward or stable trend in this plot suggests successful application of previously acquired knowledge to subsequent tasks, demonstrating positive forward transfer.

All algorithms demonstrate relatively stable and consistent performance across tasks, indicating effective forward transfer. Notably, Online EWC, despite its lower overall performance, shows exceptional stability and consistency, suggesting strong generalization capabilities. In contrast, Progressive Nets exhibit greater fluctuation, indicating that their forward transfer through lateral connections is less effective, leading to variable outcomes depending on the task. As the network expands, the features of new columns tend to become less significant overall[12,48]. TAPD consistently outperforms all other algorithms, demonstrating superior and faster task adaptability. This is evidenced by its consistently stable performance curve, which also highlights TAPD's greater sample efficiency.

The variability in the performance of Progressive Nets may stem from model complexity and the use of new lateral connections, raising concerns about its ability to consistently generalize across tasks. Online EWC distinguishes itself with stability over time and consistent performance, indicating strong forward transfer and generalization capabilities. Both Progress & Compress and TAPD offer balanced performance across tasks, with TAPD being particularly reliable for diverse tasks due to its faster adaptability. An algorithm like TAPD, which maintains low variance across tasks and visits, is demonstrably more versatile and effective at handling a variety of challenges, making it especially well-suited for environments with diverse task demands.

Computational efficiency

TAPD is a multi-phase process designed to balance computational demands with performance gains. During the initial task-agnostic phase, TAPD interacts with the environment for approximately 300,000 timesteps per task (game environment), with the total number of timesteps being dependent on the number of environments (denoted by x). For example, when $x = 25$, this corresponds to a total of 7.5 million timesteps, distributed uniformly across all environments. This phase, while computationally intensive and requiring significant interaction with the environment, is manageable within the overall process. Specifically, processing 7.5 million timesteps can take approximately 80 minutes, as shown in Table 3. Although this phase is time-consuming and represents the computational bottleneck in TAPD, it lays a crucial foundation for subsequent performance improvements. It is important to note that, by design, TAPD can operate without externally specified tasks, distinguishing it from other methods that lack this task-agnostic capability.

After completing the data collection from the task-agnostic phase, TAPD compresses the policy for each task sample. This compression step, while less computationally demanding, is critical for ensuring that the learned policy generalizes effectively across tasks. In this step, TAPD computes the Fisher information matrix for the compressed policy. Although the computation involves around 100 updates with a minibatch size of 32[11] and is

Algorithm	Training phases	Time
TAPD	Progress: DemonAttack, Pong, BeamRider, SpaceInvaders, AirRaid (2.5M timesteps)+ Compress (150,000 timesteps)	$\sim$ 20 min (each visit)
	Task-Agnostic Phase	$\sim$ 80 min
	Total Time	$\sim$ 6.3 h
Progress & Compress	Progress: DemonAttack, Pong, BeamRider, SpaceInvaders, AirRaid (2.5M timesteps)+ Compress (150,000 timesteps)	$\sim$ 20 min (each visit)
	Total Time	$\sim$ 5.3 h
Progressive Nets	Total Time	$\sim$ 5.3 h
Online EWC	Total Time	$\sim$ 4.3 h
Hardware: NVIDIA GeForce GTX 1660 SUPER and Intel Core i5-10500 CPU @ 3.10GHz		

Table 3. Computational time comparison of various algorithms on NVIDIA GeForce GTX 1660 SUPER and Intel Core i5-10500 CPU. The total time includes training and other phases without any intermediate evaluations.

relatively low in computational cost, the accuracy of the Fisher matrix might not be perfect but remains sufficient for the intended purpose.

Progressive Nets generally require a high computational cost due to their architecture. Each new task involves adding new network components, which increases the computational burden and memory requirements as more tasks are added. While they may offer strong performance on individual tasks during training, Progressive Nets are severely limited by their inability to scale, with model size growing excessively with each new task. The variance in performance in test time over visits indicates instability, as discussed in section "Assessing forward transfer", which might require additional computational resources to mitigate.

Online EWC is less computationally demanding compared to Progressive Nets. It leverages the Fisher information matrix to regularize the network weights, preventing catastrophic forgetting. The computational cost primarily arises from calculating the Fisher matrix, similar to TAPD but with a simpler model.

Despite the high initial computational cost in the task-agnostic phase, TAPD provides a strong balance between computational efficiency and performance. The time invested in this phase is offset by the improved forward transfer and generalization capabilities observed in subsequent tasks. This makes TAPD an excellent choice when the goal is to maximize performance after an initial warm-up period.

Conclusion and future work

In this paper, we presented the Task-Agnostic Policy Distillation framework, which addresses catastrophic forgetting, ensures scalability across tasks, enables positive forward transfer, and facilitates learning without requiring task labels. The framework incorporates a task-agnostic phase within the algorithmic framework of Progress and Compress proposed by Schwarz et al.[13]. This task-agnostic phase uses a self-supervised prediction error as an intrinsic reward for the agent. By doing so, the agent learns without a specific reward function and does not require clear task boundaries. The task-agnostic phase can be implemented in different variations that abstract the task-agnostic phase as a process where task-agnostic policies are distilled into the knowledge base, increasing systematic exploratory behavior. The active column in the task-agnostic phase then utilizes this knowledge base, further maximizing its intrinsic reward based on the systematic exploratory behavior. This acts as a pre-training phase before downstream tasks are applied.

It has been shown that the most general variation of the task-agnostic phase improves performance by accelerating transfer learning in the progress phase for downstream tasks, surpassing all three continual learning baselines, including the Progress and Compress method, in the Atari domain. Consequently, the Task-Agnostic Policy Distillation framework has demonstrated promising results in enhancing positive forward transfer and learning in scenarios without clear task boundaries.

Our approach addresses catastrophic forgetting by using Elastic Weight Consolidation (EWC) to protect old knowledge. Positive forward transfer is achieved through lateral connections from the knowledge base to the active column. Scalability across tasks is ensured by utilizing a single policy network for retaining old knowledge of previous tasks. This makes our framework well-suited to environments with restricted memory and onboard resources. Learning without clear task boundaries is facilitated by a task-agnostic phase that encourages exploration without relying on task-specific extrinsic rewards.

While the obtained results are promising, there is potential for further improvement. This includes exploring new variations of the task-agnostic phase. It would also be valuable to analyze variations within other domains. For instance, applying the general Variation 1 to domains such as robotic tasks. Furthermore, incorporating the intrinsic reward during the progress phase on a downstream task allows for optimizing both extrinsic and intrinsic rewards in an alternating fashion. This approach would likely be especially advantageous for long-horizon robotic tasks, where balancing exploration and exploitation is challenging, as rewards for exploration may not be received immediately.

Data availability

The datasets used and/or analysed during the current study available from the corresponding author on reasonable request.

Code availability

The implementation of our TAPD algorithm, along with the baseline methods and the code to reproduce our experiments, is available at the following repository: https://github.com/wabbajack1/TAPD.

References

1. Speranza, L., Di Porzio, U., Viggiano, D., Donato, A. & Volpicelli, F. Dopamine: The neuromodulator of long-term synaptic plasticity, reward and movement control. *Cells* **10**(4), 735 (2021).
2. Sutton, R.S., & Barto, A.G. Reinforcement learning: An introduction (2018).
3. Miller, K. D., Keller, J. B. & Stryker, M. P. Ocular dominance column development: Analysis and simulation. *Science* **245**(4918), 605–615 (1989).
4. Parisi, G. I., Kemker, R., Part, J. L., Kanan, C. & Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Netw.* **113**, 54–71 (2019).
5. Machado, M. C. et al. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *J. Artif. Intell. Res.* **61**, 523–562 (2018).
6. Shwartz-Ziv, R., & LeCun, Y. To compress or not to compress–self-supervised learning and information theory: A review. arXiv:2304.09355 (2023).

7. Pathak, D., Agrawal, P., Efros, A.A., & Darrell, T. Curiosity-driven exploration by self-supervised prediction. arXiv:1705.05363 (2017).
8. Parisi, S., Dean, V., Pathak, D. & Gupta, A. Interesting object, curious agent: Learning task-agnostic exploration. *Adv. Neural. Inf. Process. Syst.* **34**, 20516–20530 (2021).
9. Sutton, R.S., McAllester, D., Singh, S., & Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Adv. Neural Inform. Process. Syst.* **12** (1999)
10. Zhang, X., Ma, Y. & Singla, A. Task-agnostic exploration in reinforcement learning. *Adv. Neural. Inf. Process. Syst.* **33**, 11734–11743 (2020).
11. Kirkpatrick, J. et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci.* **114**(13), 3521–3526 (2017).
12. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., & Hadsell, R. Progressive neural networks. arXiv:1606.04671 (2016).
13. Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y.W., Pascanu, R., & Hadsell, R. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning* 4528–4537 (PMLR, 2018).
14. Zeno, C., Golan, I., Hoffer, E., & Soudry, D. Task agnostic continual learning using online variational bayes. arXiv:1803.10123 (2018).
15. Zhao, X., Weber, C., Hafez, M.B., & Wermter, S. Impact makes a sound and sound makes an impact: Sound guides representations and explorations. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* 2512–2518 (IEEE, 2022).
16. Bellemare, M. G., Naddaf, Y., Veness, J. & Bowling, M. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.* **47**, 253–279. https://doi.org/10.1613/jair.3912 (2013) arXiv:1207.4708 [cs].
17. Rusu, A.A., Colmenarejo, S.G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., & Hadsell, R. Policy distillation. In *ICLR* (2016).
18. Czarnecki, W.M., Pascanu, R., Osindero, S., Jayakumar, S., Swirszcz, G., & Jaderberg, M.: Distilling policy distillation. In *The 22nd International Conference on Artificial Intelligence and Statistics* 1331–1340 (PMLR, 2019).
19. Watkins, O., Gupta, A., Darrell, T., Abbeel, P. & Andreas, J. Teachable reinforcement learning via advice distillation. *Adv. Neural. Inf. Process. Syst.* **34**, 6920–6933 (2021).
20. Sun, Y., & Fazli, P. Real-time policy distillation in deep reinforcement learning. arXiv:1912.12630 (2019).
21. Thorne, J., & Vlachos, A. Elastic weight consolidation for better bias inoculation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume* 957–964 (2021).
22. Draelos, T.J., Miner, N.E., Lamb, C.C., Cox, J.A., Vineyard, C.M., Carlson, K.D., Severa, W.M., James, C.D., & Aimone, J.B. Neurogenesis deep learning: Extending deep networks to accommodate new classes. In *2017 International Joint Conference on Neural Networks (IJCNN)* 526–533 (IEEE, 2017).
23. Hafez, M. B. & Wermter, S. Continual robot learning using self-supervised task inference. *IEEE Trans. Cognit. Dev. Syst.* https://doi.org/10.1109/TCDS.2023.3315513 (2023).
24. Kemker, R., & Kanan, C. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations* (2018).
25. Ven, G. M., Siegelmann, H. T. & Tolias, A. S. Brain-inspired replay for continual learning with artificial neural networks. *Nat. Commun.* **11**(1), 4069 (2020).
26. Wang, L., Zhang, X., Su, H. & Zhu, J. A comprehensive survey of continual learning: Theory, method and application. *IEEE Trans. Pattern Anal. Mach. Intell.* https://doi.org/10.1109/TPAMI.2024.3367329 (2024).
27. Rao, D. et al. Continual unsupervised representation learning. *Adv. Neural Inform. Process. Syst.* **32** (2019).
28. Madaan, D., Yoon, J., Li, Y., Liu, Y., & Hwang, S.J. Representational continuity for unsupervised continual learning. arXiv:2110.06976 (2021).
29. Ziegler, L., Zenke, F., Kastner, D. B. & Gerstner, W. Synaptic consolidation: From synapses to behavioral modeling. *J. Neurosci.* **35**(3), 1319–1334 (2015).
30. Hinton, G., Vinyals, O., & Dean, J. Distilling the knowledge in a neural network. arXiv:1503.02531 (2015).
31. Chentanez, N., Barto, A., & Singh, S. Intrinsically motivated reinforcement learning. Adv. Neural Inform. Process. Syst. **17** (2004).
32. Dayan, P. & Balleine, B. W. Reward, motivation, and reinforcement learning. *Neuron* **36**(2), 285–298 (2002).
33. Hafez, M.B., Weber, C., & Wermter, S. Curiosity-driven exploration enhances motor skills of continuous actor-critic learner. In *2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)* 39–46 (IEEE, 2017).
34. Schmidhuber, J. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats* 222–227 (1991).
35. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. *Playing Atari with Deep Reinforcement Learning.* https://doi.org/10.48550/arXiv.1312.5602 (2013).
36. Li, Y. Deep Reinforcement Learning: An Overview. arXiv. arXiv:1701.07274 [cs] (2018).
37. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning* 1928–1937 (PMLR, 2016).
38. Wu, Y., Mansimov, E., Grosse, R. B., Liao, S. & Ba, J. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. *Adv. Neural Inform. Process. Syst.* **30**, 5285–5294 (2017).
39. Schulman, J., Moritz, P., Levine, S., Jordan, M.I., & Abbeel, P. High-dimensional continuous control using generalized advantage estimation. arXiv:1506.02438 (2015).
40. Wang, L., Zhang, X., Su, H., & Zhu, J. A comprehensive survey of continual learning: Theory, method and application. arXiv:2302.00487 (2023).
41. Fox, K. & Stryker, M. Integrating Hebbian and homeostatic plasticity: Introduction. *R. Soc.* **372**(1715), 20160413 (2017).
42. Ven, G.M., & Tolias, A.S. Three scenarios for continual learning. arXiv:1904.07734 (2019).
43. Kanervisto, A., Scheller, C., & Hautamäki, V. Action space shaping in deep reinforcement learning. In *2020 IEEE Conference on Games (CoG)* 479–486 (IEEE, 2020).
44. Paszke, A. et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inform. Process. Syst.* **32**, 8024 (2019).
45. Kostrikov, I. *PyTorch Implementations of Reinforcement Learning Algorithms* (GitHub, 2018).
46. Raffin, A. et al. Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Res.* **22**(268), 1–8 (2021).
47. Lucchesi, N., Carta, A., Lomonaco, V., & Bacciu, D. Avalanche RL: a Continual Reinforcement Learning Library. arXiv (2022). https://doi.org/10.48550/ARXIV.2202.13657
48. Xu, J., Ma, J., Gao, X. & Zhu, Z. Adaptive progressive continual learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**(10), 6715–6728 (2021).

Author contributions

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-80774-8.

Correspondence and requests for materials should be addressed to M.B.H.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.